THE UNIVERSITY of EDINBURGH

Title	Binding approach to scientific data and metadata management
Author	Chen, Yin.
Qualification	PhD
Year	2012

Digitisation Notes:

- pages ii, vi, xiv,32, 86,124,198 missing from original

A Binding Approach to Scientific Data and Metadata Management

Yin Chen

Doctor of Philosophy

Centre for Intelligent Systems and their Applications
School of Informatics
University of Edinburgh

2011

Abstract

Many e-Science applications are data intensive. The advances in automation, communication, sensing and computation enable experimental science to generate data and digital objects at great speeds and volumes. Metadata emerge as a key technology to better exploit these assets. This work views the relationship between data and metadata as a 'binding', and proposes a novel approach to managing such bindings.

Firstly, an original investigation of binding-related issues is presented. A classification framework is developed to establish a way of thinking about bindings. Evidence obtained from empirical analysis reveals that a lack of explicit binding management might cause serious problems, for example, information inconsistency.

Secondly, to clarify the concepts, a formal binding model is developed. A binding-oriented approach is proposed where bindings are first-class citizens. Desirable binding properties and operations are defined.

Based on the formal model, a novel binding management approach is derived, where bindings are independently managed. The binding representation and system operations are specified. The viability of the design is demonstrated by a prototype implementation. The performance of the binding system is quantified by experimental measurements. To provide scalability, cloud computing technology is adopted and experimental investigation shows a Hadoop-based binding system is capable of processing very large numbers of bindings.

Finally, a proof of concept for the binding approach is provided by a practical installation in a concrete scientific application. The experimental results confirm that the binding approach is useful in facilitating such systems at affordable computational costs. In addition, the use of binding in a set of use scenarios demonstrates the value of this approach in support of a variety of scientific applications.

Although the study has been carried out in the context of e-Science applications, the approach is much more widely applicable.

Acknowledgements

I would like to thank my supervisor Professor Malcolm Atkinson, for his guidance and the wise advice he gave me throughout my PhD study. Thanks to my second supervisor Dr Stuart Aitken, for all the beneficial discussions. Great thanks to Professor Don Sannella, Professor Ewan Klein who have reviewed my study and gave many advice and useful information. Thanks to Professor Wenfei Fan for the discussions of the formal model.

I am grateful to people and organisations for providing access to their resources:

- National e-Science Centre;
- Professor Richard Baldock, and Dr Duncan Davidson from MRC HGU;
- Professor Susan Lindsay, and Steven N. Lisgo from Newcastle HDBR;
- Professor Peter Burnhill, Chris Higgins and Dr David Medyckyj-Scott from EDINA Data Library and Geo Research.

Much help and support has been obtainted from the following people and projects, great thanks to them:

- Data resources are obtained from: Dr Clive Davenhall and Professor Richard Sinnott from the NanoCMOS project; the DGEMap project; and the EurExpress project;
- Technique supports are obtained from the OGSA-DAI team;
- Research materials are obtained from Professor Carole Goble and Dr Oscar Corcho from the OntoGrid project; Dr Philip Lord from the MyGrid project; and the COBrA-CT project.

Thanks the University of Edinburgh for providing the scholarship.

Finally, thanks to my close friend Jean Walker, my dear parents and sister. Thanks very much for their consistent encouragement and support. Their love carries me throughout long journey.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Yin Chen*)

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

This work examines the problems of scientific data and metadata management from a new point of view. Many e-Science applications are data intensive. Throughout the UK and the world, huge quantities of data and digital objects are gathered at great expense. Metadata emerges as a key technology to better exploit these assets by allowing data to be published with their associated contextual information so as to aid their future discovery and reuse. This work views the relationship between data and metadata as a 'binding', and investigates approaches to support the management of such bindings as first class entities[1].

The need to efficiently manage the data-metadata relationship arises in a variety of scientific applications, such as a provenance system which tracks the biological workflows, or a distributed medical image annotation system. In such applications, the data and metadata are often created and stored separately, e.g., they may be generated by different users, in different computing processes, stored at different locations, in different types of storage. Data and metadata may have different lifecycles, e.g., they might have been created at a different time, updated and removed independently. Often, there is more than one set of metadata related to a single data resource, e.g., when the existing metadata becomes insufficient, users may design new templates to make another metadata collection. Without efficient software and tools, binding management becomes onerous.

A biological high-throughput image generation and annotation system, the EurExpress project[2], was studied. Numerous problems were identified, such as information inconsistency, and traced to the lack of an effective binding mechanism. The empirical evidence shows that bindings between data and metadata are vulnerable to failures in the processes that create and maintain them, and to failures in the systems that store their representa-

[1] Informally, a binding can be thought of as a link or association between data and metadata. See section 2.1.3 for a discussion of why we have chosen the term 'binding'.

[2] EurExpress project: www.eurexpress.org

tions. As bindings carry semantic information that may only exist by virtue of the binding, it is imperative to devise methods that reduce these failures, ideally to zero.

While the distinction between data and metadata is generally accepted, and many metadata schemes are in use, no reported architecture is known where these bindings are explicitly managed. The primary motivation for this work is to make a first attempt to a practical implementation of a binding system to manage scientific data and metadata.

1.2 Hypothesis and Challenges

This work evaluates the hypothesise that:

> *A simple binding system, which stores and manipulates the binding representations between data and metadata is both feasible and useful, in combination with other services and tools, for serving the various types and scales of scientific data and metadata in a distributed computing context.*

In the studies of existing scientific applications we observed that bindings were embedded within other parts of the data or metadata representation. We hypothesis that this had not been a considered design choice but a minimal effort as the need arose incrementally during system development. As the bindings were not explicitly exposed, their subsequent management become difficult or infeasible. Our hypothesis is that by identifying bindings and providing a binding service as easy to use as Flickr[3], the future developers of scientific applications would find use of this approach to be the minimal effort path. This would result in clearer binding semantics and facilitate subsequent binding management and evaluation.

There are many challenges. Problems in the management of data-metadata bindings exist, but they have not been clearly identified as yet. Very few discussions about bindings can be found in literature. Existing systems and applications are designed for various purposes, and describe their approaches from traditional (or their own) points of views. What is needed is to develop the understanding of binding requirements through observations, establish the way of thinking about the new concept, examine the existing systems from binding point of view, and identify any related problems.

Having improved the understanding of the bindings, it is still difficult to design a generic tool to manage various types and scales of scientific data and metadata. The diversity of scale and representations of scientific data and metadata presents significant challenges. Bindings should have existence independent from data and metadata. The computational representation for bindings stored and manipulated by a binding system should be simple and tolerant of semantic diversity of various data and metadata, meanwhile it should be able to support scalable data/metadata processes. However, as there is

[3]Flickr: http://www.flickr.com/, a popular web service for users to manage and share photos.

little prior experience of handling bindings independently, design decisions on a suitable data structure for bindings are difficult to make. In addition, a binding system runs in distributed environments where data and metadata resources are pre-existing autonomous storage systems and beyond the control of the binding system. It is a design challenge to specify appropriate binding operations to support complex dynamic behaviours of scientific applications. Such operations should be easy to understand and efficient to use.

As the binding system is the first implementation of such a system, it is difficult to evaluate the feasibility of the approach and convince users to adopt the new tool.

During the research exploration, these questions will be answered. The next section provides an overview of our approach.

1.3 Our Approach

To develop an understanding of bindings, we start from the observations and analysis of real-world systems. A classification framework is developed, and used to analyse characteristics of bindings in various computational contexts. An empirical study of a chosen use case reveals numerous binding-related problems. Then a formal model is provided to clarify the concept and to capture the binding requirements obtained from real-world observations. The formal model is used as a foundation allowing the binding problems identified from real practices to be described precisely.

The key contribution of this work is a novel binding approach to scientific data and metadata management. Based on the formal model, a simple binding system is proposed which manages bindings as independent entities between data and metadata. The proposed binding systems consists of a binding data structure where both data and metadata elements are references to the respective sources, and bindings have user-assigned tags that describe the nature of the binding. A set of operations for managing the binding are specified. The design of the binding system is kept as simple as possible. This can largely reduce implementation cost, and avoid complexity when integrating with external tools or services. The intention is to allow easy adoption by application developers.

The binding model is implemented as a web service. This allows user applications to associate web-accessible data and metadata, and to annotate the contents and the contexts of the bindings by using tags. The binding service is not designed for scientists to use directly, rather it is intended to serve as a functional unit in conjunction with other services and tools, for example, to act as a third end-point between the existing data resources and metadata resources and provide binding facilities. In this way, the binding service can extend existing systems to have binding facilities automatically. The binding system presents the following features:

The system stores the references to the data and metadata rather than the physical datasets. Traditionally, the distributed data and metadata have often been copied and housed together in a centralised repository. For example, the EurExpress project pools the gene expression images and their annotations from partner biology labs, and the Avian Knowledge Network (AKN)[4] uses a data warehouse to collect and store all 60 distributed bird occurrence data sets and their metadata [Kell 09]. However, the data-metadata warehousing approach is quickly becoming infeasible, as more data and metadata sets are becoming too large, too dynamic or just too unwieldy to be co-located productively. In some cases the owners of the data or metadata may not permit copies. While the proposed approach has the potential disadvantage that to retrieve data and metadata may involve multiple requests and hence be time consuming, it has the advantage that all data and metadata no longer need to be copied to a central server and will be up-to-date with the original data sources. The proposed approach also has the advantage that it does not require co-operation from data and metadata owners, except for stability of references. In addition, it can list data and metadata held in multiple sources.

Several systems provide an ad hoc binding mechanism within their system framework. For example, the OntoGrid project[5] provides a Semantic Binding Service to create, store, update and remove the linkages of the Grid Entities and Knowledge Entities consumed by other system components [Corc 06, Gobl 05]. Nevertheless, the OntoGrid's Semantic Binding Service is designed to address the specific issue of adding semantic descriptions to Grid service of the OntoGrid system. It is not designed for general data-metadata management. In contrast, the proposed approach provides a generic binding model, which is widely applicable to various e-Science applications. In a sense, the OntoGrid Semantic Binding Service can be regarded as an implementation instance of the binding model proposed. Another related work is the Linked Data approach[6] which addresses many similar issues to the binding service. However, the design of the two approaches is different in many ways. A comparison (which will be presented in Chapter 8) shows that the binding service approach has restricted behaviours in data referencing which can significantly reduce system complexity.

In addition to the comparative analysis in Chapter 8, the performance of binding service is evaluated and usage of this approach is demonstrated. A workload modelling and simulation method is used to quantify the performance of the binding service. The characteristic workloads of creations and queries that such a service must support are reported. So that, synthetic workloads with comparable properties can be simulated. Then, the efficiency of the binding service for handling simulated workloads are measured. To

[4] AKN: www.avianknowledge.net

[5] OntoGrid project: www.ontogrid.net

[6] The Linked Data approach: http://linkeddata.org/

demonstrate usage of the binding service, a sample installation of the binding service to the EurExpress system is provided, which implements a validation operation to check inconsistency between EurExpress data and metadata. The experimental analysis results show that the binding service is able to detect realistic binding failures in the existing systems at affordable computational cost. To illustrate binding usage in other types of scientific applications, scenario-based evaluation is applied, and binding solutions to seven representative use scenarios are provided.

1.4 Thesis Structure and Contributions

This section describes the organisation of the thesis arguments and highlights the contributions of the research.

Chapter 2 contributes an original investigation of binding-related issues in various scientific systems. This provides evidence for the requirements for efficient binding management. The observations, which start with scientific data and metadata phenomena, are then scoped into metadata centric applications and further narrowed down to a specific domain application and a typical use case. The binding classification framework is described. Typical binding patterns in five classes of scientific applications are reported. From the empirical study of the EurExpress System, significant unsolved binding problems are identified which poses open research questions for e-Science.

Chapter 3 contributes a survey of related approaches from the literature. Representative scientific systems managing data and metadata are examined from a binding point of view. The chapter also explores a number of state-of-the-art data-reference approaches, and technologies that could be used in design or implementation of the binding system.

Chapter 4 contributes a first formal model for bindings which defines the concepts and establishes a formal foundation to discuss the binding problems. A binding-oriented approach is proposed, where bindings are the first-class citizens and have an existence independent of their associated data and metadata. Necessary binding properties and operations are defined.

Based on the formal foundation, a novel design of binding service is proposed, which implements a data-reference model. **Chapter 5** describes the design including the specification of the binding data structure and a set of binding operations. The viability of the design is demonstrated by a prototype implementation.

The functionality and efficiency of the binding service is evaluated experimentally in **Chapter 6**. Characteristic workloads are reported by analysing a number of existing scientific systems. In addition, Cloud technology is explored to provide scalability properties to the system, and the performance of a Hadoop-based binding storage system is quantified by empirical measurements.

Chapter 7 contributes a sample installation of binding service in a real-world application, and the value of the binding service in supporting that application is evaluated experimentally. Chapter 7 also contributes binding solutions to seven use scenarios which illustrate that binding approach can be wildly applied to support various types of scientific application.

Chapter 8 evaluates the binding approach by comparing it with a related approach, the Linked Data model. The chapter contributes a detailed comparison of the two approaches from five perspectives including, data structure, binding expressions, storing mechanisms, search capabilities, and communication capabilities.

Chapter 9 concludes the study and reviews the discoveries. It also discusses the aspects for improvements and directions for future research. Although the study has been carried out in the context of e-Science applications, we believe our contributions are generally applicable to many data management systems beyond e-Science applications.

Chapter 2

Understanding Binding

Modelling and managing relationships between computational objects has led to many fruitful results in Computer Science. While the distinction between data and metadata is generally accepted, and many metadata schemes are in use, no explicit definition of bindings and their architectural characteristics have been reported. This chapter presents original investigations of binding related issues, examining the questions what is binding, why is it important, and how to identify a binding in use, so as to establish a way of thinking about bindings. The observations, which start with scientific data and metadata phenomena, are scoped into metadata-centric applications, and further narrowed down to a specific domain application and a typical use case.

The chapter is organised as follows: observations on scientific data and metadata phenomena are presented in section 2.1. A binding classification framework for analysing metadata-centric scientific applications is developed in section 2.2. A study of a selected use case appears in section 2.3, and section 2.4 provides binding scenarios. Finally section 2.5 summarises this chapter.

2.1 Observations of Scientific Data and Metadata Phenomena

2.1.1 Scientific Data

Many scientific applications are data intensive. The advances in automation, communication, sensing and computation enable experimental scientific processes to generate data and digital objects at great speeds and volumes. The Sloan Digital Sky Survey (SDSS) catalogue over the first 5 years has collected roughly 290 million objects (4TB in size), and after 10 years contained 12TB of data [Ahma 10]. The world biggest telescope, Pan-STARRS, at the end of the first year, contained 70 billion detections in 30TB, and at the end of the project will grow to several petabytes [Szal 09, Hay 09]. The Laser Interferometer Gravitational Wave Observatory (LIGO) system stores more than 40 million files

across ten locations. The two LIGO detectors at Livingston and Hanford produce data at a rate of slightly less than a terabyte per day during LIGO experimental runs. Each detector produces a file every 16 seconds that contains data for those 16 seconds of measurements. These files range in size from 1 to 100 MB [Cher 05]. More examples: the Australian SKA Pathfinder (ASKAP), a 1% Square Kilometre Array pathfinder radio telescope, generates ~10TB data per hour [Guzm 10]. LOFAR, the low frequency array, is building a huge radio interferometer. The data stored in LOFAR will rise from 800TB data in 2010 to 1.5PB in 2011, and the total data volume will rise up to 20PB in the next 5 years of observations [Bege 11].

Another remarkable characteristic about scientific data is that they are heterogeneous objects without unity of representation. Different community groups produce a range of diverse data type such as proteomes, gene expression, protein structures, and pathways. The data covers different scales and different experimental procedures. The various databases and tools have different formats, access interfaces, schemas, and coverage. They typically have different, often home grown, versioning, replication, authorisation, and provenance policies.

These two characteristics of scientific data present significant challenges for managing, reusing, and sharing the digital data collections. To exploit the growing wealth of scientific data, there has been a commensurate growth in the creation and use of metadata. Metadata allows data to be published with their associated information, so as to aid their future discovery and reuse. Without descriptions, the content and context of the scientific datasets cannot easily be understood by another user or computer system.

2.1.2 Scientific Metadata

Metadata, the so-called data about data, is used in scientific applications to describe, explain, locate, or make it easier to retrieve, use, or manage an information resource. The term 'meta' comes from a Greek word that denotes 'alongside, with, after, next'. In philosophy, 'meta' is used as a prefix to denote an alternate or second-order kind of relationship between two similar types of entities and the underlying notion of the essential attributes that make up a metadata description [Gill 08]. Since the 1960s, bibliographic metadata, such as library metadata standard, MARC (Machine-Readable Cataloging format), LCSH (Library of Congress Subject Headings), and AAT (Art & Architecture Thesaurus), have been co-operatively created and made available to repositories and users through automated systems [Gill 08]. Until the mid-1990s, metadata was a term most used by geospatial communities, and data management and system design and maintenance in general. Over recent years, the concept of metadata has been popular on the World Wide Web, especially in the Semantic Web, to aid online document and resource discovery.

Table 2.1: Scientific Metadata Syntax Examples

Standard	Full Name	Organisation
HTML	Hyper-Text Markup Language[1]	The World Wide Web Consortium (W3C)
XML	eXtensible Markup Language[2]	The World Wide Web Consortium (W3C)
RDF	Resource Description Framework[3]	The World Wide Web Consortium (W3C)
OWL	Web Ontology Language[4]	The World Wide Web Consortium (W3C)
SGML	Standard Generalised Markup Language[5]	The World Wide Web Consortium (W3C)
MARC	Machine Readable Cataloging[6]	Library of Congress
MIME	Multipurpose Internet Mail Extensions[7]	Internet Engineering Task Force (IETF)
DIME	Direct Internet Message Encapsulation[8]	Internet Engineering Task Force (IETF)
netCDF	network Common Data Form[9]	Unidata, the University Corporation for Atmospheric Research

There have been numerous attempts to classify the various types of metadata. As one example, NISO (National Information Standards Organisation) distinguishes between three types of metadata based on their functionality: *Descriptive metadata*, which describes a resource for purposes such as discovery and identification; *Structural metadata*, which indicates how compound objects are put together; and *Administrative metadata*, which provides information to help manage a resource [NISO 04]. But this is not restrictive. Different applications may have different ways to classify their own metadata.

Metadata is generally encoded in a *metadata schema* which defines a set of metadata elements and the rules governing the use of metadata elements to describe a resource. The characteristics of metadata schema normally include: the number of elements, the name of each element, and the meaning of each element. The definition or meaning of the elements is the *semantics* of the schema, typically the descriptions of the location, physical attributes, type (i.e., text or image, map or model), and form (i.e., print copy, electronic file). The value of each metadata element is the *content*. Sometimes there are *content rules* and *syntax rules*. The *content rules* specify how content should be formulated, representation constraints for content, allowable content values and so on. And the *syntax rules* specify how the elements and their content should be encoded. Some popular syntax used in scientific applications are listed in Table 2.1. Such syntax encoding allows the metadata to be processed by a computer program.

[1] HTML: http://www.w3.org/MarkUp/
[2] XML: http://www.w3.org/XML/
[3] RDF: http://www.w3.org/RDF/
[4] OWL: http://www.w3.org/2001/sw/
[5] SGML: http://www.w3.org/MarkUp/SGML/
[6] MARC: http://www.loc.gov/marc/
[7] MIME: http://www.ukoln.ac.uk/metadata/resources/mime/
[8] DIME: http://xml.coverpages.org/draft-nielsen-dime-01.txt
[9] netCDF: http://www.unidata.ucar.edu/software/netcdf/

Table 2.2: Scientific Metadata Standards Examples

Standard	Full Name	Organisation
Dublin Core	Dublin Core Metadata Initiative [10]	Dublin Core Metadata Initiative
ISO 19115	ISO 19115:2003-Geographic Information–Metadata [11]	ISO/TC 211
CSDGM	The Content Standard for Digital Geospatial Metadata[12]	Federal Geographic Data Committee (FGDC)
DDI	Data Documentation Initiative Metadata Specification[13]	Documentation Initiative
INSPIRE	Infrastructure for Spatial Information in Europe[14]	European Commission (EC)
SEED	Standard for the Exchange of Earthquake Data[15]	U.S. Geological Survey, IRIS
VOEvent	Sky Event Reporting Metadata[16]	International Virtual observatory Alliance (IVOA)
OAIS	Reference Model for an Open Archival Information System[17]	The Consultative Committee for Space Data System (CCSDS)
EML	Ecological Metadata Language[18]	National Center for Ecological Analysis and Synthesis (NCEAS)

Many standards for representing scientific metadata have been developed within disciplines, sub-disciplines or individual project or experiments. Some widely used scientific metadata standards are listed in Table 2.2.

Two aspects of metadata give rise to the complexity in management:

Metadata are data, and data become metadata when they are used to describe other data. The transition happens under particular circumstances, for particular purposes, and with certain perspectives, as no data are always metadata. The set of circumstances, purposes, or perspectives for which some data are used as metadata is called the 'context'. So metadata are data about data in some 'context' [Stan 04].

Metadata can be layered. This happens as data objects or information resources may move to different phases during their life in a digital environment, thus requiring layers of metadata that can be associated.

In both cases, the relationships between metadata and information objects, and between different aspects of metadata, should be reliably managed.

2.1.3 Binding of Scientific Data and Metadata

When (a set of) metadata is associated with its data object (or resource), a relation between them is generated. We use term '***binding***' to denote each data to metadata association. (We use term 'binding' rather that other terms such as, 'link' or 'association', to distinct from

[10] Dublin Core: http://purl.oclc.org/metadata/dublin_core/
[11] ISO 19115: http://www.iso.org/iso/catalogue_detail.htm?csnumber=26020
[12] FGDC–CSDGM: http://www.fgdc.gov/standards
[13] DDI: http://www.ddialliance.org/
[14] INSPIRE: http://inspire.jrc.ec.europa.eu/index.cfm/pageid/101
[15] SEED: http://www.geoinstr.com/pub/manuals/t_seed.pdf
[16] VOEvent: http://www.ivoa.net/Documents/latest/VOEvent.html
[17] OAIS: http://public.ccsds.org/publications/archive/650x0b1.pdf
[18] EML: http://www.loc.gov/standards/mods/

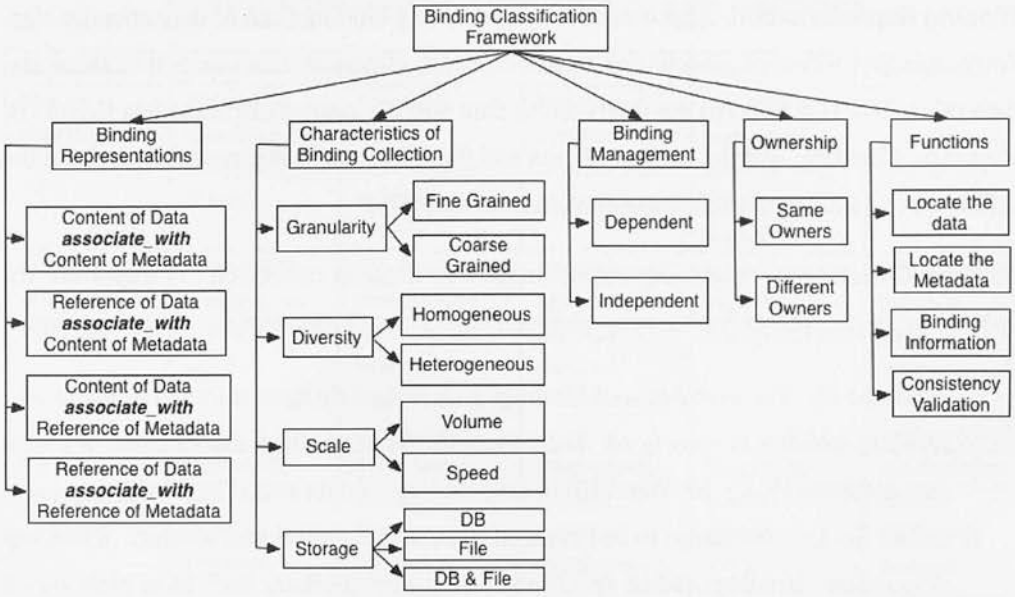


Figure 2.1: Binding Classification Framework

existing concepts. For example, 'link' is used by the Linked Data as a term to describe the specific approach which exposes, shares and connects data (information or knowledge) on the Semantic Web. The term 'association' has been used in various situations, which may easily mislead readers to think about a different contexts. In a later chapter, a formal definition of binding will be provided.)

It is evident that binding is a meaningful notion consisting of characteristic features. However, in practice, the concept of binding is not well established, and bindings exist in various forms. It is necessary to clarify issues unresolved in the literature and develop an understanding as to what a binding is, and why it is important. An investigation of bindings in real world scientific applications is presented below.

2.2 Observations of Bindings in Scientific Applications

In order to establish a way of thinking about bindings, a classification framework is developed in an attempt to create a generalised taxonomy which is useful for identifying characteristics of bindings in use, and with the view of informing future research. The classification framework is built on an observation and analysis of five classes of metadata-centric scientific applications, including: (i) scientific annotation management, (ii) long-term archive and curation, (iii) information integration, (iv) workflow and service discovery, and (v) provenance management. As shown in Figure 2.1, the framework provides a feature set consisting of five perspectives.

Binding Representation. Approaches for representing bindings can be conceptually classified into four forms of association: (i) the contents of data and the content of the metadata [Kava 01, Tuff 06]; (ii) the reference to data and the contents of metadata [Myer 03, Raja 02b, Cher 05]; (iii) the content of data and the reference to the metadata, and (iv) the reference to data and the reference to metadata [Gobl 06].

Binding Collection. Analysing characteristics of binding collections is important for specifying requirements for bindings management. Four characteristics are considered:

1) *Granularity.* The usefulness of bindings in a certain domain is linked to the granularity at which it is associated. The requirements range from associations of tuples in a database [Yang 10, Yama 10] to associations of data sets [Corn 04].
2) *Scale.* Scales of bindings to be managed relate to the speeds and volumes of binding generation. Bindings, when specified at finer granularities, such as in provenance applications [Geha 10, Fost 02], can grow exponentially in the number of recorded steps. On the other hand, bindings, when in the form of content and content associations, can be large in size [Weil 06].
3) *Diversity.* This is to look at content differences and structural variations of bindings data and metadata, which are heterogeneous or homogenous. Information contents can be uni-disciplinary or multi-disciplinary, binding data and metadata can be standardised or non-standardised. In terms of diversity, a binding collection can contain (i) uni-disciplinary and standardised data and metadata associations [Yang 10, Yama 10]; (ii) uni-disciplinary and non-standardised data and metadata associations [Geha 10]; (iii) multi-disciplinary and standardised data and metadata associations [Prli 07]; and (iv) multi-disciplinary and non-standardised data and metadata associations [Saye 10].
4) *Forms of storage.* Different applications use different storage systems to store bindings. Some systems store binding as database entries [Yang 10, Yama 10, Prli 07], some store bindings in files [Weil 06, Raja 02b], and some others use hybrid methods [Saye 10].

Binding Management. Many existing systems manage bindings through a *dependent* model where the distributed metadata (and data) have been copied and housed together in a centralised repository. For example, the EurExpress systems pools the gene expression images and their annotations from other biology laboratories, and the Avian Knowledge Network (AKN) uses a data warehouse to collect and store all 60 distributed bird occurrence data sets and their metadata, which has taken considerable effort [Kell 09]. However, the data-metadata warehousing approach is quickly becoming infeasible, as more data and metadata sets are becoming too large, too dynamic or just too unwieldy to be

Table 2.3: Characteristics of Bindings in Scientific Applications

Application	Annotation	Archive	Integration	Workflow	Provenance
Binding Representation	Annotation *associate_with* coordinator of data	Content of metadata *associate_with* id of dataset	Reference of metadata *associate_with* reference to data	Reference of metadata *associate_with* reference to workflow	Provenance *associate_with* id of process or data
Characteristics of Binding Collection	Finer grained, homogeneous, large scale, stored as database entries	Coarser grained, homogeneous/ heterogeneous, large scale stored as database entries or files	Coarser grained, heterogeneous, large scale, stored as database entries	Coarser grained, stored as description files	Finer grained, large scale, standardised metadata, stored as database entries
Binding Management	Dependently managed through annotation	Dependently managed through metadata in catalogues	Independently managed through registry service	Dependently managed through web service	In a distributed context, bindings are managed independently
Ownership	Same/different owners	Often under the same ownership	Same/different owners	Same/different owners	Often under the same ownership
Functions	Linking annotations with data object. Locate data items of the annotation	Locate the dataset of the metadata	Linking various metadata and data sources	Validating workflow plan, linking workflow with metadata	Linking provenance with processes

copied productively. In some cases the owners of the data or metadata may not permit copies.

An alternative approach to binding management is an *independent* model where data and metadata are managed through references. While an *independent* binding model has the potential disadvantage that to retrieve data and metadata may involve multiple requests and hence be time consuming, it has the advantage that all data and metadata no longer need to be copied to a central server and will be up-to-date with the original data sources. It also has the advantage that it does not require co-operation from data and metadata owners, except for stability of references. In addition, it can list data and metadata held in multiple sources.

Ownership. Concerning the ownership of binding data and metadata, we distinguish two situations: 1) both binding data and metadata under the same ownership; and 2) the binding data and metadata under different ownership, for example, many scientific systems are built on collaborations across different organisations, where the binding data and metadata belong to single or multiple owners, respectively.

Functions. Bindings can be used in various ways, for example, to locate the data and metadata; to maintain the consistency between data and metadata; to link data with mul-

tiple metadata sources; to record provenance and statistic information related to data and metadata; to link different version of data and metadata; and to support metadata-driven workflow.

Table 2.3 gives a summary of the analysis of bindings in various scientific metadata-centric applications. In the following, binding characteristics in each domain applications, as defined by the classification framework, are discussed in turn.

2.2.1 Scientific Annotation

In every area of science, investigations now depend not only on new experiments, but also (and perhaps more frequently) on databases in which experimental evidence has been accumulated. This evidence is seldom raw experimental data; it is typically some form of interpretation of the data, and annotations are increasingly important parts of those interpretations [Bose 06a]. Annotation metadata are comments, notes, explanations, or other types of external remarks that can be attached to an existing base or target that possesses structure [Bose 06b].

In biology, gene annotation databases are widely used as public repositories of biological knowledge. For example, deepBase contains annotations of ~380,000 unique ncRNA-associated small RNAs, ~1.5 million unique promoter-associated small RNAs, ~4.0 million unique exon-associated small RNAs and ~6 million unique repeat-associated small RNAs [Yang 10]. DBTSS (Database of Transcription Start Sites) includes ~330 million annotations for transcription start sites (TSSs) [Yama 10]. Many bioinformatics systems, such as PROSITE [Sigr 10], SIMAP [Ratt 10], and Gene3D [Lees 10], employ DAS (Distributed Annotation System)[19], a lightweight decentralised system, to exchange and aggregate annotations from a number of heterogeneous database [Prli 07].

In these bioinformatics systems, annotations are statements about the structures and functions of genes, and bindings are the mappings of annotations to the genome regions or gene IDs. The mappings are often generated by similarity search algorithms, such as BLAST [Moun 04] and Bowtie [Lang 09], which align the sequence reads to the chromosomes. The computational representation of the binding takes the form of the association of the annotation identifier and the sequence reference. The sequence reference can use any coordinate systems, e.g., deepBase and DBTSS use the start- and stop- point of the DNA on the chromosome, and DAS uses the entry point and the length of DNA on the chromosome.

Genome annotation is meant to consistently capture the scientific community's progress in understanding the way genes function and represent the most recent knowledge about

[19]DAS specification: http://www.biodas.org/documents/spec.html

the genes of various organisms. To be useful, it needs to integrate diverse resources to ensure a wide enough collection. For example, deepBase includes the ~11.8 million annotations from 185 small-RNA Libraries, and the DBTSS integrates the ~300 million TSS tags from 31 different libraries.

The contents of genome annotation databases are subject to change. Because of the dynamic nature of biological studies, each year the limit of detection is greatly increased, and new annotations or updated annotations continue to emerge. The definition of a finished genome is necessarily a moving target [Schl 05]. Take the DBTSS project as an example. With the updated knowledge that a large number of human genes turn out to contain multiple promoters, a simple catalogue of the promoters as often provided in the pre-existing database becomes insufficient to represent a global view of transcriptional regulation in human genes [Yama 10]. The endeavour of the DBTSS is to generate new TSS tags of eukaryotic mRNAs by sequencing the 5'-end of oligo-cap selected cDNAs in diverse cellular environments [Yama 10].

Annotation bindings represent a characteristic pattern of binding collections. In the annotation-binding pattern, data and metadata tend to be associated within finer granularities, for example, binding metadata can be a statement or a list of tags, and binding data can be a coordinate in an n-dimensional space. The bindings are large in number, generated by many independent organisations, and in many situations, bindings have multiple data owners and multiple metadata owners. The representations of the bindings tend to be simple and uniform, and the contents of bindings can be changed overtime.

The annotation-binding pattern is also found in the Sloan Digital Sky Survey system, where bindings are the associations of descriptive tags with the coordinates of the astronomical objects on the surface of the sphere [Szal 00b]; in a Geographic Information Systems (GIS), the DBpedia Mobile [Beck 09], where the background information of places are bound to the locations represented in an earth coordinate system; in a biomedical image annotation systems, the Allan Brain gene expression database [Jone 09], where gene expression annotations are bound to the coordinates of the anatomical regions on the 3D atlas; and a distributed bookmark system, the Delicious[20], where descriptive tags are bound to the URLs of website objects.

To support the annotation-binding pattern, a system should provide lightweight and flexible annotation mechanisms, as well as support large-scale sharing of bindings.

2.2.2 Scientific Curation and Archive

Curation aims to maintain digital research data and other digital materials over their entire life-cycle for current and future generations of users [Beag 06]. Curation includes

[20]Delicious: http://delicious.com/

the process of digital archiving and preservation, and the capacity to add value to data to generate new sources of information and knowledge. Examples of curation and archive databases include the Protein Data Bank (PDB)[21], which archives biological macromolecular crystal structures; the NASA Life Sciences Data Archive (LSDA)[22], which preserves information and data from NASA's space flight experiments, and the Planetary Data System (PDS)[23], which provides archiving service and data access for NASA-funded planetary science community. The curation/archive projects often exist for a long time, e.g., the LSDA started since 1961[22], PDB since 1971[24], and PDS since 1993[23]. The range of data types (e.g., for different ways of determining structure) and the range of uses have changed significantly over this time.

Bindings in curation/archive systems often exist in a coarse granularity, and in two main forms: the form of association of the metadata record with the identifier of the archive dataset, which are often stored in the metadata catalogues; and the form of association of the metadata file and the dataset file, which are stored in the archive media.

Binding data in many curation/archive databases are stored in uniform formats, e.g., PDB specify the submission data format, the so-called PDB format. And the LSDA preserves all data on CD-ROMs in the ISO9660 format[25] and all data are written into ASCII. But there can be exceptions, e.g., PDS stores data as their original format to enable continue access by the existing software. During submission, an archive dataset is usually given a unique identifier internally by the archive system.

Binding metadata in archive systems tend to be standardised. Archive systems usually define metadata standards and provide metadata editors allowing professional curators to describe submitted datasets, i.e., PDB and PDS use templates, and LSDA manages metadata through the FileMaker. The standards would change when new data types appear, and software may also be changed or updated. Each metadata record is usually stored in catalogues for discovery, and a file copy is saved with the dataset to be archived together, so that the catalogues can be reconstructed.

Bindings in archive systems may grow in proportion to the accumulated data which often starts at a slow pace and then accelerates quickly. For example, at the beginning the PDB archive held only seven protein structures, and in the 1980s the number of deposited structures began to increase dramatically. In 2001 there were over 16,500 entries in the archive [Berm 02], as of this writing in December 2010, this number is nearly 70,000[26].

---

[21] PDB: http://www.wwpdb.org/

[22] NASA LSDA: http://lsda.jsc.nasa.gov/

[23] NASA PDS: http://pds.nasa.gov/

[24] According to Helen M. Berman, in his article, *The Protein Data Bank: a historical perspective*, 2007.

[25] ISO9660 is an ISO published file system standard for optical disc media.

[26] PDB statistics: http://www.pdb.org/pdb/static.do?p=general_information/pdb_statistics/index.html, retrieved Dec. 2010

Successful long-term high quality data preservation requires maintenance of the consistency of bindings between data and metadata to reduce the likelihood of important data becoming incomprehensible and un-usable over substantially long periods of time. Mechanisms should be provided to efficiently register the bindings, and dependably manage them.

2.2.3 Scientific Information Integration

Conventionally, an individual researcher develops hypotheses, designs experiments to test these hypotheses, collects observational data, and publishes results based on experiments. Nowadays, much scientific research is achieved by widely distributed, multidisciplinary collaborations which enable scientific discovery based on evaluating multiple competing hypotheses using multiple types of evidence and relating new findings to the existing knowledge and literature in a field. Information integration emerges as the key technology for seeking to maximally exploit available information to create new scientific knowledge.

Take one of the e-Science domain practices as an example: the Nucleic Acids Research January 2010 includes descriptions of 58 new and 73 updated data resources, its online database collection lists 1230 carefully selected databases covering various aspects of molecular and cell biology. These data collections are far more than a distinct set of resources; they form a virtual knowledgebase of connected data, concepts and shared technology. Besides archive databases, a large number of integration databases exist. The archive databases, such as the Protein Data Bank (a protein structures resource) [Vela 10], the European Nucleotide Archive (ENA, the Europe's primary nucleotide sequence archival resources) [Lein 10], the GenBank (a nucleic acid sequence database) [Bens 08], ELM (a eukaryotic linear motif resource) [Goul 10], the European Mouse Mutant Archive (EMMA, a mouse mutant strains resource) [Wilk 10], and the PubChem BioAssay Database (an archive of small molecules and small interfering RNAs) [Wang 10], typically collect and maintain specific biological domains' experimental descriptions and information, with links of references and related information from other databases. The integrated databases are built upon these resources; they combine data from various archival databases and reorganise them to form new views of knowledge. For example, the National Center for Biotechnology Information (NCBI) resource provides analysis and retrieval resources from the GenBank, the Entrez, the PubMed, the EntrezeGene and other biological data made available through the NCBI web site [Saye 10]. ChimerDB identifies genes from bioinformatics analysis of transcript sequences in the GenBank, OMIM, PubMed and Sanger cancer genome project [Kim 10]. The Comsin database from the complete PDB selects pairs of protein structures in bound and unbound states and identifies regions of intrinsic disorder [Loba 10], while the PDBselect selects

representative protein chains with low mutual sequence identity from the PDB [Grie 10].

To have a close look at how bindings are used in the integration systems, take the NCBI resource as an example, which integrates a variety of databases, applications and tools related to the molecular biology. Entrez [Saye 10, Schu 96] is an integrated database within NCBI that provides access to a diverse set of 38 databases that together contain over 400 million records. Entrez supports downloading of data in various formats and linking of records between databases based on biological relationships. Although the concepts of data and metadata are ambiguous here, the Entrez links can be regarded as bindings in the context of this investigation. Entrez links exist in various forms. In the simplest forms, they may be cross-references between a sequence and the abstract of the paper in the PubMed, or between a protein sequence and its coding DNA sequence or its 3D structure [Saye 10]. In a more complex form, there are computationally derived Entrez links between 'neighbouring records', such as those based on computed similarities among sequences or among PubMed abstracts, which allow rapid access to groups of related records [Saye 10]. On the other hand, a service called LinkOut expands the range of Entrez links to include external services (owned by other organisations), such as organism-specific genome databases. The linked datasets can be displayed in many formats and downloaded singly or in batches.

Entrez Gene of NCBI [Magl 06] provides descriptive information about more than 5.4 million gene and sequences with bindings to NCBI's Map Viewer, Evidence Viewer, Model Maker, Blink, protein domains from the NCBI Conserved Domain database (CDD) and other gene-related resources, and bindings to the newest citations in PubMed.

The Molecular Modelling Database of NCBI [Wang 07] contains experimentally determined coordinate sets from the Protein Data Bank, attached with bindings to domain annotations, relevant literature, protein and nucleotide sequences, chemicals and conserved domains in CDD, and structural neighbours in the 3D Domains database.

PubMed database of NCBI contains more than 19 million citations dating back to the 1860s from more than 21,000 life science journals [Saye 10]. Over 10 million of these citations have bindings to their full-text articles [Saye 10]. PubMed is heavily linked to other core Entrez databases where it provides a crucial bridge between the data of molecular biology and the scientific literature. PubMed records are also linked to one another within Entrez as related articles on the basis of similarity algorithms.

Bindings in the integration applications tend to be diverse, and many of them use references to represent the associations between data and metadata. One of the challenges for supporting integration-binding pattern is how to provide a unified binding model and system representation to include the information of various types from extant data and metadata. In principle, such binding model should be simple and extensible.

2.2.4 Scientific Workflow

Scientific problem solving is an evolving process. Scientists start with a set of questions then observe phenomenon, gather data, develop hypotheses, perform tests, negate or modify hypotheses, reiterate the process with various data, and finally come up with a new set of question, theories or laws. Most of the time before this process can end in results, scientists will fine-tune their experiments, going through much iteration with different parameters [Alti 06]. Scientific workflow represents the logical abstraction of this process.

Scientific workflow is the application of workflow technology to scientific endeavours. It aims to capture the necessary semantics of computational resources, and provide robust problem-solving environments that enable bindings of any scientific data and services. Currently, the (Semantic) Web Service and Service Orientated Architecture (SOA) technology have been popularly adopted in this paradigm.

Surveys on scientific workflow systems have been provided in literature [Deel 09, Curc 08], the following representative workflow systems are selected for analysis of bindings.

The ^{my}Grid middleware framework provides web services for biological experimental analysis [Hull 06]. ^{my}Grid executes workflows written in $XScufl$ language using the Taverna engine. Each Taverna workflow can have metadata stored inside it using the author and title tags. In this case, bindings are in the form of embedded metadata within the workflow. Additional workflow metadata can be stored separately from the workflow and identified using a Life Science Identifier (LSID). All workflows have an LSID by default, and metadata are assigned to this LSID [Hull 06]. In this case, bindings exist as the associations of the reference to workflow and the reference to metadata.

Kepler [Alti 06, Alti 04], is a scientific workflow construction, composition, and orchestration engine, and capable of modelling processes in a wide variety of scientific domains, from physics to ecological web services. The workflow components in Kepler are called *actors* which have input ports and output port that provide the communication interface to join another *actor*. Data and explicit metadata are represented as a sequence of *tokens* which are passed from one *actor* to another. Bindings of data and metadata are managed through the Kepler actors, which are in the forms of a sequence of tokens.

Wings [Kim 06, Deel 09] provides a semantic metadata generation and reasoning approach that supports creation of large workflows. Wings' metadata describe constraints on files and collections, written in OWL-DL and stored as XML files. A Wings' workflow template is an abstract specification of a workflow with a set of nodes and joins, where each node is a placeholder for iterative execution of a program over a file collection, and each connector represents how the input and output parameters are connected. Given a metadata file and a workflow template, Wings validates the workflow plan using

the metadata constraints. To do this, Wings traverses joins in the workflow template and generates consistent bindings for joins, which binds files or collections, and the metadata constraints. The output of validation is an executable workflow plan with all joins attached to valid bindings.

Bindings in workflow applications are normally embedded in the workflow constructions to reference data and metadata sets. To validate a workflow plan, it often involves the examination of the availability of data and metadata. This observation of workflow-binding pattern suggests that it would be useful for a generic binding management system to provide operations for checking the availability and accessibility of binding data and metadata.

2.2.5 Scientific Provenance

The scientific processes demand the precise tracking of how a particular result has been achieved, in order to be able to explore many different alleys simultaneously, moving forward to well defined states when successful, or rolling back otherwise. Provenance awareness becomes important under this requirement.

Provenance captures the derivation history of a data product, starting from its original sources [Simm 05]. It makes possible for independent replication of experiments, location and recovery from faults, and ensures that work is repeatable. Example provenance systems include Chimera [Fost 02], ESSW [Frew 01], and SPADE [Geha 10].

Chimera is a virtual data catalogue that provides representation of the workflow procedures used to derive data (the so-called *transformation*), the invocations of those procedures (the so-called *derivation*), and the datasets produced by those invocations [Fost 02]. A Chimera *derivation* can be regarded as a binding, which associates the name of a Chimera *transformation*, and the names of the data objects to which the *transformation* was applied. A data object is a named entity that may be consumed or produced by a *derivation*. A data object is always a logical file, named by a logical file name. A separate replica catalogue, or replica location service is used to map from logical file name to physical locations for replicas. When executing, workflows automatically create *derivation* objects for each *transformation* in the workflow, annotated with runtime information of the process. The *derivation* objects, which are the bindings, link process input and output data products, and they constitute an annotation scheme for provenance metadata.

ESSW, the Earth System Science Workbench, is a metadata management and data storage system for earth science researchers [Frew 01]. ESSW tracks how science objects are associated, and the origin and processing history of each science object – the object's provenance – is collected [Frew 01]. Bindings in ESSW exist as the links of science objects, representing the to-from relationship between science objects; they are stored in

a database table, ScienceObjectLink, which allows the attachment of their provenance. ESSW provenance provides information to move backwards through linked inputs, processing steps, and outputs, which enables a data-user to discover the source of errors. Descending a line of connected objects can identify erroneous products.

SPADE is a distributed provenance system to manage provenance across hosts and organisational boundaries [Geha 10]. The main challenge for SPADE is how to audit the movement of files so that there is no loss of coupling between the file contents and the associated provenance metadata [Geha 10]. Bindings exist as SPADE provenance graphs, which associate files with their execution processes. In the SPADE provenance graphs, files are identified by the pathname in the host's filesystem, the size of the file, the last time it was modified and a hash of its contents. Execution processes are identified by the process name, operating system identifier, owner and group. *Edges* of the SPADE provenance graphs associate an input file with the process, and the process with an output file.

Managing provenance-binding patterns requires low-overhead binding generations, and scalable strategies to store and search bindings which are potentially huge in volume and increase quickly.

2.2.6 Conclusion

In this section, a classification framework has been introduced to understand and conceptually describe characteristics of bindings. Sample systems from five metadata-centric e-Science application domains have been analysed from binding point of view. Although binding management is still an exploratory field with many and varied applications, the widespread need to support the (creation and use of) bindings is clear. Table 2.4 summarises the usage of bindings in various applications.

Table 2.4: Examples of Binding Usage

Application Domain	**Uses of Bindings**
Genome Annotation	Associate annotations to sequence reference
Life Science Archive	Maintain data-metadata consistency
Bio-information Integration	Link and reference information
Metadata-driven Workflow	Locate data/metadata
Distributed Provenance	Record data-metadata provenance

The investigation illustrate that in many scientific applications the data and metadata were often created and stored separately: they may be generated by different users, in different computing processes, stored at the different locations, in different types of storage. Data and metadata may have different lifecycles, i.e., they may have been created at

a different time, updated and removed independently. Often, there is more than one set of metadata related to a single data resource. For example, when the existing metadata becomes insufficient, users may design new templates to make another description. Without efficient software and tools, binding management becomes onerous. In fact, a lack of explicit and effective binding management may cause serious problems. The evidence was obtained from an empirical analysis as described below.

2.3 Observations of A Binding Use Case

To identify binding-related problems, a modern biological information system, the EurExpress project, has been investigated. Metadata are heavily used for image management in EurExpress. The project uses advanced robotics technology, GenePaint [Vise 04, Bola 06], to enable high-throughput generation of biological images, and provides the means for scientists to observe biological tissues or cells via the virtual microscope, an image browser. Metadata describe the content, background knowledge, and processing instructions of the images. Without metadata, biologists would be less informed by the digital objects that originate from an observation, experiments, or computation. As most of the existing information retrieval systems or image processing software are still text-based, metadata are crucial to enable applications to interact with images, to provide human- and machine-interpretable information, e.g., to find images associated with a particular gene expression, a particular stage of development or a particular part of the anatomy.

2.3.1 The EurExpress Project

The sequencing of the genomes of various organisms, such as the fruit fly, mouse and human is one of the major recent successes in biology. This achievement leads to the next great goal of understanding the physiological role of a given gene and the proteins it codes for, and how they collectively interact to carry out cellular processes, in particular, to determine when, where, and at what cell type, a (subset of the) gene is expressed. To do this, biologists use a technique known as in situ hybridisation (ISH), to collect the experimental data, which combines molecular biological techniques with histological and cytological analysis of gene expression, and is capable of revealing gene expression in individual cells within their natural tissue environment. ISH on serial sections cut through the tissue of interest, when appropriately spaced, provide the required resolution and data quality [Jin 97]. Nowadays, genome labs all over the world, design and perform ISH experiments on different growth and development of species, ranging in complexity from insects to humans, and generate vast volume of gene expression data [Dunl 08].

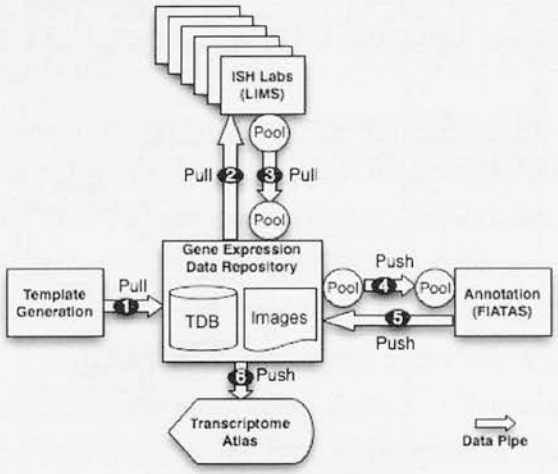


Figure 2.2: EurExpress System Infrastructure

The goal of EurExpress project is to capture the expression patterns of ~20,000 genes via ISH. These are recorded from sagittal histological sections from 14.5 days wild type mouse embryos, and result in a digital transcriptome atlas. Each has a detailed description of gene expression patterns in terms of the underlying anatomy.

Generation of the gene expression data is co-operatively performed by 11 biological laboratories, across 6 European countries. It involves three main processes: a) A biological laboratory in Germany responds to select 20K genes and create experiment protocols (so-called templates), and probes for the ISH; b) The prepared templates are distributed to the partner laboratories, where the high-throughput robots automate the ISH process and produce the high quality gene expression images; and c) The gene expression images are grouped by gene into assays and sent to a biological laboratory in Alicante for annotating.

The system infrastructure is depicted in the Figure 2.2. At its centre is the Gene Expression Data Repository (GEDR) located in Edinburgh Medical Research Council (MRC) Human Genetics Unit (HGU), consisting of the Tracking Database (TDB) and a file system. Communication channels are established between the GEDR and computational components located in other laboratories. A spreadsheet application is used by the experiment design team to generate the templates in the Excel file formats. A LIMS system is installed at each ISH lab to collect the experimental datasets and images. In the EurExpress project, each ISH experiment typically generates 24 images of embryo sections, ~200MB in total, with the corresponding metadata in XML format. The ISH result datasets and images are sent to the GEDR through an FTP channel. The GEDR converts the metadata from XML files into database entries; and the images are processed and stored in the file system of the GEDR. Each gene expression image is assigned a globally unique identifier, with syntax as '`euxassay_[0~9]*6`'. Each metadata entry is also given

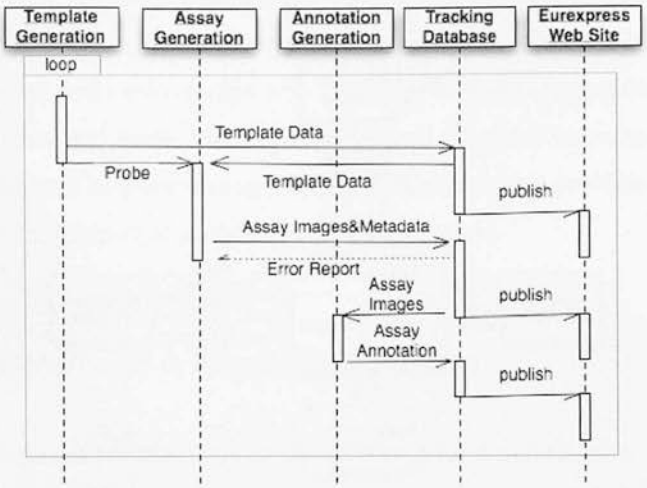


Figure 2.3: EurExpress System Process Sequence

a globally unique identifier. The identifier of an image and the identifier of the metadata are stored as a binding in the database. The Fast Image Annotation Software (FIATAS) is used in the Alicante lab for biologists to annotate the images. The completed annotations are stored in a copy as XML files in Alicante, and transferred to the GEDR. The GEDR converts them into database entries.

The entire process sequence of the system is described in Figure 2.3. When the templates are generated (in batches), the GEDR is notified to open the channel to obtain the template and store them in the TDB (also see step 1 in Figure 2.2); The GEDR then notifies the partner laboratories to access the template and start the ISH experiments (step 2 in Figure 2.2); The ISH laboratories submit the experiment results, the gene expression images and the metadata, to a shared data pool. When the data pool is filled up, the GEDR opens the communication channel to pull the data in. Error data unable to be processed by the GEDR, are reported back to the originator (see step 3 in Figure 2.2). Then, the GEDR wraps the gene-expression images with necessary metadata and sends them for annotation (step 4 in Figure 2.2). The annotations once completed are delivered to the GEDR via web services at run time. A monitor service is setup to continuously capture the abnormal operations and report to the system administrator (step 5 in Figure 2.2). The gene expression data once prepared are published through a web portal, and the research community can view the transcript atlas immediately (step 6 in Figure 2.2).

After running for two years, by July 2008, the system has collected 19,717 template records, 15,414 assays records, totally 4TB gene-expression data in store. Among them, 11,825 assays have been manually annotated by the biologist experts. By June 2010, the system comprised 18,000 assays, and 80% of which have been annotated [Han 11].

2.3.2 Empirical Analysis of EurExpress Bindings

Definition of Bindings in Use

In the context of the EurExpress system, the file system of the GEDR is regarded as a data Set, $\boldsymbol{D_g}$, which stores the gene expression images, $\boldsymbol{d}$. Related descriptive information is regarded as the metadata, $\boldsymbol{m}$, of a gene expression image. This includes, the template record, $\boldsymbol{m_t}$ in Set $\boldsymbol{M_t}$, the ISH experiment record $\boldsymbol{m_e}$ in Set $\boldsymbol{M_e}$, and the annotation $\boldsymbol{m_a}$ in Set $\boldsymbol{M_a}$. The metadata sets, $\boldsymbol{M_t}$, $\boldsymbol{M_e}$, $\boldsymbol{M_a}$, are submitted to the GEDR by the distributed biological laboratories. Each biological laboratory holds a local copy of $\boldsymbol{M_t}$, $\boldsymbol{M_e}$, $\boldsymbol{M_a}$, denoted as $\boldsymbol{M_t}$', $\boldsymbol{M_e}$', $\boldsymbol{M_a}$', which are stored in local file systems. Each individual association of $\{\boldsymbol{m_t}, \boldsymbol{d}\}$, $\{\boldsymbol{m_e}, \boldsymbol{d}\}$, $\{\boldsymbol{m_a}, \boldsymbol{d}\}$, $\{\boldsymbol{m_t}', \boldsymbol{d}\}$, $\{\boldsymbol{m_e}', \boldsymbol{d}\}$, and $\{\boldsymbol{m_a}', \boldsymbol{d}\}$ is defined as a binding, respectively.

Method of Data Collection

The following binding relationships are examined:

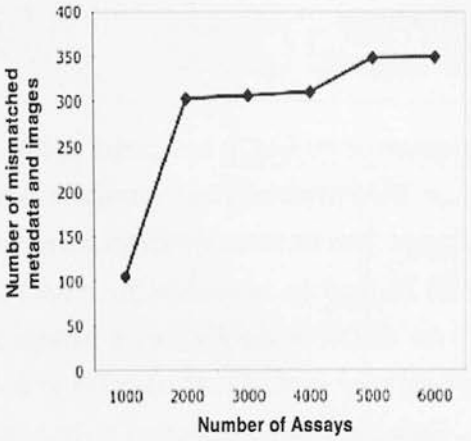
Case 1: $\boldsymbol{M_e}$ with $\boldsymbol{D_g}$. The aim is to examine the binding status in simple metadata and data association processes. Each entry of the metadata database, $\boldsymbol{M_e}$, and the gene expression images in the file system, $\boldsymbol{D_g}$, are examined. They should exist in matched pairs.

Case 2: $\boldsymbol{M_t}$, $\boldsymbol{M_e}$ with $\boldsymbol{D_g}$. The metadata Set $\boldsymbol{M_t}$ is produced by the template generation which is a pre-process of ISH. The ISH process produces the metadata Set $\boldsymbol{M_e}$, and $\boldsymbol{M_e}$ uses some information from $\boldsymbol{M_t}$, i.e. the gene symbols. The aim is to examine the binding status when one piece of metadata information is transferred from one process to another. To do this, the overlapped information, the gene symbols, are separated from, $\boldsymbol{M_t}$, and $\boldsymbol{M_e}$, and the contents of them are compared. The information from the two sets should be exactly the same.

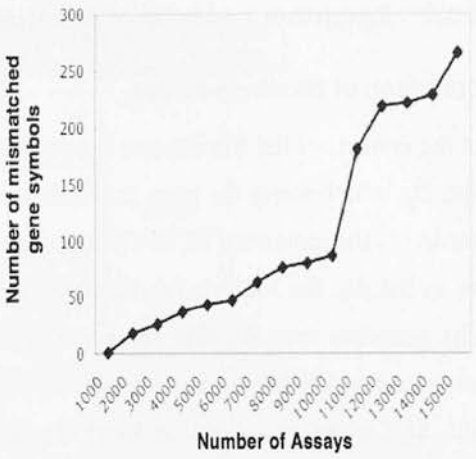
Case 3: $\boldsymbol{M_a}$, $\boldsymbol{M_a}$' with $\boldsymbol{D_g}$. This case aims to examine the binding status when metadata exist in multiple copies and in different computational forms. $\boldsymbol{M_a}$ is the annotation set stored in the database of the GEDR, and $\boldsymbol{M_a}$' is the same annotation information stored in the file system in the Alicante lab. Entries in both copies are compared, and the relationships of $\boldsymbol{M_a}$, $\boldsymbol{M_a}$' with $\boldsymbol{D_g}$, are examined, respectively.

Analysis Results

Results for Case 1. Recall that the ISH-generated gene expression data, images and their metadata, were grouped by gene into 'assays', and delivered to the GEDR. The GEDR stores the images in the file system, and inserts the image metadata into the database table. However, the simultaneously processed information pairs are found to be different in number. Graph (a) in the Figure 2.4 shows the cumulative numbers of the mismatched pairs – images without metadata and metadata without images. The x-axis presents the numbers of assays. The y-axis presents the cumulative numbers of mismatches. Among



(a) **The numbers of mismatched metadata and images**



(b) **The numbers of mismatched gene symbols**

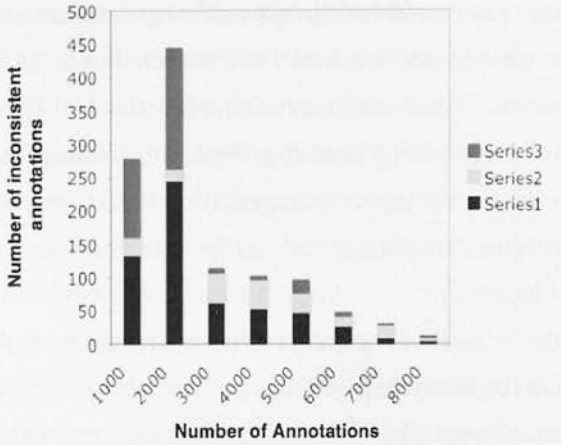
Figure 2.4: Analysis Results for Case 1 and Case 2 of EurExpress

the first 2000 assays, a total of 315 mismatches were detected. Digging for the reason, it was discovered that the ISH quality checking has been carried out after the gene expression data had been stored. It detected wrongly generated or low quality of images. Most of the missing metadata were of those disqualified images. They have been manually deleted from the database, but (for some reason) those disqualified images have not been removed from the file system. (A plausible explanation: it might due to the project policy that the ISH image generation were too expensive. Though faulty, the images might become useful later.) The reason that the metadata exist but not the images, however, remains unknown.

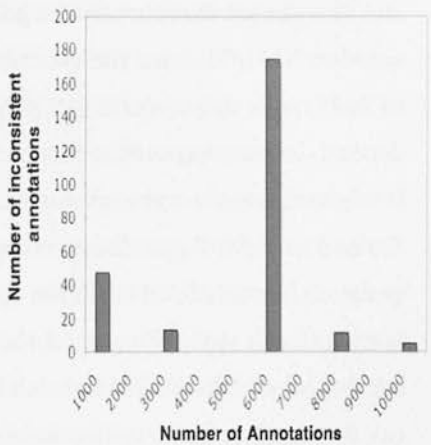
After the jump at 2000, these errors level off. This appears to be due to the change of data management policy. The metadata of the faulty images were no longer deleted from the database, only marked as invalid by using a flag value. The figures reflect the improvement of the information consistency. On the other hand, the lack of an explicit binding that can be interrogated easily means that there is a potential hazard if automated processes are applied to the set of assay directories or to the image files – data already known as being of inadequate quality might easily be included in the process and might deleteriously affect the quality of the derived data.

Results for Case 2. Graph (b) of Figure 2.4 shows the results of the investigation of Case 2. The x-axis refers to the numbers of assays. The y-axis refers to the cumulative numbers of mismatched gene symbols.

In total, 1.76% gene symbols used in ISH do not match with the same information in the design templates. It is difficult to determine to which gene the expression stain generated by the ISH related. The reason for this was that the updates of the template



(a) **The numbers of annotations in the file system but not in the database (series1), the numbers of duplicated annotation files (series2), and the numbers of invalid annotation files (series3)**



(b) **The numbers of annotations in the database but not in the file system**

Figure 2.5: Analysis Results for Case 3 of EurExpress

information were not reflected the information used in the later process. Lacking systematic management, the updates could not be propagated automatically. This results in an amount of mismatched information.

Another observation is that the number of erroneous matching occurrences grows linearly with the numbers of assays, and does not level off. This shows the information related via the template design tends to be changed often, and is difficult to make stable. Consequently, there will always be a proportion of incorrect matches unless the management of these bindings is improved to enable better automation.

Results for Case 3. (1) Recall the annotation of gene expression images was performed at one of the partner laboratories at Alicante. The biologists use the FIATAS to annotate the images, which sends the completed annotations to the GEDR, and stores a local copy in XML format. Graph (a) of Figure 2.5 illustrates that the numbers of annotation XML files in Alicante's server that do not match the numbers of annotation records in the GEDR. (The x-axis refers to the numbers of annotations. The y-axis refers to the numbers of the inconsistent annotations. The numbers of the inconsistent annotations are calculated in each batch of 1000 entries.) In the Graph (a):

- Series 1 shows the numbers of annotation XML files existing in the Alicante's file system but not in the GEDR.
- Series 2 shows the numbers of duplicated annotation XML files (in different folders).
- Series 3 shows the numbers of invalid annotation XML files related to the deleted assays.

The sum of the three values gives the total number of discrepancies between the annotation XML files and the database records for each batch of 1000 entries. In the 2000 to 2999 range this amounts to a 45% error rate, which will inevitable affect the fidelity of derived information and the complexity of subsequent processing. Analysing the reasons, it might due to the network connection failures during the transportation (in the case of Series 1); might due to disagreement or redone annotation work (in the case of Series 2); or due to human mistakes (in the case of Series 3).

(2) Graph (b) of Figure 2.5 shows the numbers of annotation information present in the database of the GEDR but absent from the file system at Alicante. The same as Graph (a), the x-axis refers to the numbers of annotations. The y-axis refers to the numbers of the inconsistent annotations, which are calculated in each batch of 1000 entries. The reason of the inconsistency remains unknown. From the graph it can be observed, this random behaviour is uncorrelated with entry number, i.e., the length of time the project has been underway and the maturity of the software and operational procedures. This suggests that these forms of failure cannot be eliminated simply by improving the system – hypothesise is that they are intrinsic to multi-party, multi-site binding creation and maintenance.

2.3.3 Conclusion

Bindings are vulnerable to failures in the processes that create and maintain them, and to failures in the systems that store their representations. The failure of the bindings may seriously impact on data retrieval in a number of ways, for example:

- It effects the precision and recall of data objects in response to a query on the metadata; Relevant data may be overlooked, on the other hand, irrelevant data may be returned; Data retrieval may return multiple pointers to the same data;
- Data interpretation – the units, experimental protocol and other contextual elements may be missing or in error; and
- May cause the wrong information combination in provenance or authority.

Without efficient management, the information inconsistency will grow and propagate widely, subsequently, the expensively generated data will lose their value quickly, and the analysis and research based on them will be untrustworthy.

2.3.4 Discussion

In addition to the problems detected in the EurExpress system, other types of inconsistency are identified through a literature survey and experience. They can be grouped into five categories:

Inconsistency in Identifying and Recognising the Data Objects. Data objects and data resources (the variety of data) are chosen by and supplied by end users. These data items need to be identified and recognised by a consuming system. Internally, there may be computational identifiers defined by the system, such as full path names in file systems. Externally, the users may use a short form assuming context or some other identifier established by their operational practices. A reliable bi-directional mapping is required between these two forms of identification. Problems arise when this bi-directional mapping develops failures, such as: non-uniqueness of external identities; the loss of an internal identifier (and perhaps its identified data item) though the external identifier is in use; the loss of an external identifier corresponding to an internal identifier (and data item).

Inconsistency in mapping the metadata to the data items. That is, the binding has been assigned to the wrong data items. The Case 1 investigation in the EurExpress described above falls into this category.

Inconsistency in the dimension of time. During their lifetime in the system, the data and metadata can be modified or transferred from one form to another form. Consequently, different versions coexist in different processes causing confusion [Aitk 08, Aitk 07a, Aitk 07b]. Strategies should be introduced to propagate the updates in both up and down directions along the process chain, and allow tracking of the changes to the digital objects. Three sub-classes can be further discriminated in this category:

- Mutation of the data and metadata values. That is, updating to either the stored data object or the stored metadata values or to the binding's representation.
- Changes in the representation (format or encoding) of the values. This is common as improved engineering delivers better representations or as standards are adopted or revised.
- Changes in the structure (types and schemata), which often reflects advances in understanding by the teams organising components of the system. These changes are continuous and fundamental. No amount of investment in agreements will stop them. They reflect the progress that the research (and the technology that supports it) is achieving.

Inconsistency in the dimension of space. Often, many copies of data and metadata items coexist, with the replications distributed over different sites, servers and in different types of storage system. Much research has been conducted in this area, i.e. various replication services are provided by the Grid applications [Kuns 03, Cher 05, Nich 06, Lame 02, Cigi 05, Bell 02, Limi 05, Chen 07], and the relational database replication approaches provided by Oracle [Garm 03], Sybase [Clif 95], DB2 [Gu 02, Cicc 04], MySQL

[Zawo 04], Microsoft SQL Server [Shar 02]. However, open questions remain in the context of scientific data-metadata, including:

- Are special, efficient algorithms and representations available for handling the replication of the set of data-metadata bindings?
- Does the use of distributed replication for the data objects have to use logical naming if they are represented as references in the system. Is a special form of logical naming appropriate?
- Does the use of distributed replication for metadata warrant any special treatment?
- There are already good distributed transaction algorithms. Could they be applied, especially, when some of the sites are not prepared to make changes to their systems? and
- Are there affordable and practical ways of introducing redundant values, c.f. checksums and digital signatures, to support binding-error detection?

Inconsistency in semantics and syntaxes. People describe things differently, using different terms and formats. A single concept may be interpreted differently when the definition or description are ambiguous. i.e., the following problems are detected:

- *Date*, e.g., in the EurExpress system, date expression such as '08/04/05' were difficult to interpret, which could be either '08/Apr/2005', 'Aug/04/2005', or '2008/Apr /05';
- *Unit*, e.g., the failure of the Mars Climate Orbiter on 23rd September 1999 was caused by the program wrongly interpreting the thrust requirement instructions in pounds force – the SI 'metric' unit, which should be in Newtons (one pound force is approximately equal to 4.45 newtons). This resulted in the probe inserted at a mere 57 kilometres altitude instead of 150 kilometres, and burned up in the Martian atmosphere [Ober 99].
- *Price*, this is distinguished from the 'unit' here, since the 'prices' problem is a special case arising within the financial databases and their (web) applications, i.e., in some web applications, the users from different countries input the financial data based on their own living context, which results in the price of the product in the database table being mixed up with pounds, dollars, and euros .
- *Other types of ambiguity*, e.g., in the EurExpress system, the gene expression image 'set' has been interpreted as 'the total set', 'the set No.', 'the set Id' in different biological labs.

These 5 categories of inconsistency problems are very hard to resolve. Some of them remain as open questions for future work. However, an easily used binding framework

should ameliorate many of these difficulties and provide a platform on which to build more complete solutions.

2.4 Binding Scenarios

Having illustrated the importance of explicit binding management, this section provides characteristic binding scenarios which, in essence, are the simple sketches of the applications discussed above:

Binding Scenario 1: A researcher is creating files and associating a few tags to each that may be used later to select files for some task.

Binding Scenario 2: The researcher has collected a large number of directories of files with their metadata poorly encoded in their names. Now he wants to use the binding store to reorganise access independent from directory trees and to access relevant subsets.

Binding Scenario 3: The researcher wants to use files of their own as in Scenario 1 or Scenario 2 plus files in a reference repository and select subsets from the combination.

Binding Scenario 4: The researcher has done Scenario 1, Scenario 2 and Scenario 3, now wants to share some (or all) of his data with his collaborators.

Binding Scenario 5: A group of collaborators want to share descriptions including referenced data.

Binding Scenario 6: Any the above but the referenced data are subsets of the data in a database or parts of one or more files.

Binding Scenario 7: The collaborators have built a valuable collection of bindings of data and metadata, and now want to make that public for other researchers.

The seven scenarios starting from simple and basic requirements, incrementally build up the complexity for the design of binding systems. Keep these pictures in mind, later chapters will review them in numerous occasions.

2.5 Summary

This chapter has investigated the binding related issues. The understandings were gained through three steps of observations: the observations of scientific data and metadata phenomena, the observations of scientific metadata-centric applications, and the observations of selected use case. The challenges for managing binding have been identified, which will be further explored in later chapters. The next chapter discusses related work.

Chapter 3

Related Work

This chapter presents a review of other research related to this thesis, and an introduction of technologies used in design and building the binding system. The first section provides a survey on existing binding management approaches. Section 3.2 looks at data-reference models. Section 3.3 introduces tagging and Cloud technology, and how researchers have explored these areas. Finally section 3.4 summarises this chapter.

3.1 Binding Management Approaches

The binding classification framework given in Chapter 2 has distinguished two binding management approaches: (1) the *dependent* binding management where bindings are in the form of associations of values, and the distributed metadata (and data) are copied and hosted together; and (2) the *independent* binding management where bindings are in the form of associations of references, and metadata and data are managed through those references. Some systems have been mentioned as examples. This section looks at the details of the sample systems plus some other comparable systems. The aim is to expose how bindings are managed and to examine the data structures and behavioural operations.

Five representative *dependent* binding management systems and one *independent* binding management system are selected. Among the *dependent* binding management systems, (1) MCAT is a metadata catalogue designed for Data Grids[1]; (2) UDDI is a web service registry standard[2]; (3) Feta is a web service registry designed for a special domain of science; (4) BIRN is a Semantic Grid[3] system designed for integrating a variety of

[1] Data Grids are Grid systems which focus on enabling the creation of shared collections from data distributed across administrative domains, and support data intensive applications [Moor 08a]. Example systems include, SRB, iRODS and SHAMAN.

[2] After initial popularity UDDI has largely been abandoned. We introduce UDDI approach here to give an example of early practices in metadata management.

[3] Semantic Grids are Grid systems which focus on reasoning of attributes inferred from logical relationships between semantic terms [Moor 08a]. Example systems include BIRN, GEON, NEON, SEEK,and caBIG.

biomedical data resources and applications; and (5) gLite AMGA is a generic metadata service designed to serve different types of (Grid) applications. In contrast, OntoGrid is one of few systems implementing an *independent* binding management approach.

3.1.1 A Simple Metadata Catalogue Approach – MCAT

Storage Resource Broker (SRB) system[4], developed by Reagan Moore's[5] team at the San Diego Supercomputer Centre, is a Data Grid middleware that provides a uniform interface to access heterogeneous distributed storage resources, such as file systems, database systems and archival storage systems [Moor 06]. MCAT is SRB's metadata catalogue, which keeps track of namespaces and mappings of data objects to storage resources within a federation.

Data Structure

MCAT stores the information of the physical location of a given data object, file attributes, metadata (mainly resource locations and types), access control lists, storage resource information, and user data. Internally, MCAT handles three levels of metadata [Raja 02a]: (1) Digital object metadata, which is created for every data collection for supporting information discovery; (2) System-level metadata, which is used to provide location transparency, access transparency and protocol transparency; and (3) Schema-level metadata, which is used to federate data collections and to migrate the collection to a new technology. Among them the schema-level metadata keeps all of the attributes that are defined, including (i) Logical Structure, which is used for accommodating registered metadata; (ii) Attribute Clusters, which are the ontology for the terms in the attribute domain; (iii) Token Attributes, which are used to capture simple semantic information; and (iv) Linkages, which are the bindings, and used for interoperating within and between schema. Externally, MCAT defines an interface protocol, which uses a special data structure, the Metadata Attribute Presentation Structure (MAPS), to provide a uniform structure for communicating between MCAT servers and user applications [Raja 02a].

MCAT is implemented within the SRB infrastructure. SRB presents clients with a logical view of data sets stored in the SRB. Each data set stored in SRB has a logical name. The physical location of the data set is mapped to the logical name. The actual data of a data set belonging to the same collection may physically reside in different storage systems. The mapping, in other words, the binding, is stored as the metadata associated with the data set in the MCAT [Moor 06, Raja 03].

[4] SRB: http://www.sdsc.edu/srb/index.php/Main_Page

[5] Reagan with some of his team has moved to the East Coast in North Carolina (RENCI) and had a new technology, iRODs, with many of SRB's properties. iRODS will be introduced in section 3.2.

Behavioural Operations

MCAT uses a query generator to access internal and external metadata catalogues. It supports SQL queries for the relational database catalogue, and LDAP-specific queries for the LDAP catalogue. MCAT defines four SQL-syntax-style data structures for querying and retrieving metadata, and they are: `MAPS_Query_Struct` (used in querying the meta catalogue), `MAPS_Result_Struct` (for transferring results from the catalogue), `MAPS_Update_Struct` (for ingesting and modifying meta information in the meta catalogue), and `MAPS_Definition_Struct` (for defining and exchanging meta information schemata) [MCAT 10].

SRB/MCAT provides an integration solution by implementing the logical name space and managing the bindings of logical name and physical attributes, therefore the changes of resources would not impact on managing identity and integrity metadata [Moor 06]. However the MCAT bindings, either the associations of data and metadata or the mappings of logical name and storage resources, are not separately maintained rather stored and manipulated together with the MCAT metadata. This is potentially problematic, i.e., when metadata is inaccessible, binding information will be inaccessible. Moreover, the representations of MCAT bindings are specified to associate data and metadata within SRB systems, and inapplicable to a broader computational context. Next, we look at an approach that can associate external data and metadata.

3.1.2 A Web Service Registry Approach – UDDI

Universal Description, Discovery, and Integration (UDDI)[6], a W3C recommended web services registry standard, defines a Web-based distributed directory mechanism that enables a wide range of businesses to list themselves on the Internet and discover each other.

Data Structure

UDDI defines four data entities [OASI 04]: (i) `businessEntity` which stores contact information about the organisation that published the service; (ii) `businessService` which stores the description of a service's business function; (iii) `bindingTemplate` which stores the service's technical details, including a reference to the service's programmatic interface or API, and (iv) `tModel` which stores various other attributes or metadata such as taxomony, transport protocols, and digital signatures. The UDDI data model can be regarded as a binding representation which associates the contents of service metadata (`businessEntity`, `businessService`, and `tModel`) with the reference of service (`bindingTemplate`).

UDDI data model presents a hierarchical structure. A `businessEntity` element contains a `businessServices` element which is a container for `businessService` elements;

[6]UDDI: http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm

a businessService element contains a bindingTemplates element, which is a container for bindingTemplate elements. The bindingTemplate entries point to tModel entries [OASI 04].

UDDI provides a classification mechanism which is based on three predefined taxonomic categorisation and classification schemes [OASI 04]: (1) the North American Industry Classification System (NAICS) for classifying businesses by industry; (2) the Universal Standard Products and Services Classification (UNSPSC) for product and service classifications, and (3) ISO 3166 standard for geographic location classifications. When they register, businesses are requested to provide the relevant classification information which are stored in tModel [OASI 04].

Behavioural Operations

UDDI defines two types of operations: (1) the Publication APIs which are used for authentication (including get_authToken, discard_authToken); and (2) the Inquiry APIs which are used for query and manipulation, including *find* operations (find_business, find_service, find_binding, find_tModel), *get* operations (get_businessDetail, get_ serviceDetail, get_bindingDetail, get_tModelDetail), *save* operations (save_business, save_service, save_binding, save_tModel), and *delete* operations (delete_business, delete_service, delete_binding, delete_tModel) [OASI 04].

UDDI is mainly designed for business domain applications, when serving scientific domain applications, the hierarchical data structure is ineffective to support scalability and semantic diversity. These issues have been indicated and addressed by other research, such as Feta.

3.1.3 A Semantic Web Service Registry Approach – Feta

*my*Grid is a large UK e-Science pilot project providing open source high-level Grid middleware to enable a virtual workbench for data-intensive bioinformatics [Gobl 03]. One of *my*Grid's key products, Taverna[7], provides a workflow tool enabling scientists to orchestrate bioinformatics web service and existing bioinformatics applications in workflows [Oinn 04]. Feta[8] is Taverna's first service registry and provides semantic discovery service within the context of *my*Grid.

Data Structure

Binding representation in Feta is defined by the *my*Grid data model which includes three main entities: service, operation, and parameter [Lord 05].

[7]Taverna: http://www.taverna.org.uk/

[8]Feta: http://www.mygrid.org.uk/tools/service-management/feta/

The `service` entity encapsulates information relating to publication, including the provider organisation name, the author of the service description, and a free-text description of the functionality.

Attributes of `operation` include: the task being performed by the operation; the method being used; the application to which the service belongs; and the resource that the service uses.

The inputs and outputs of an operation are modelled through the `parameter` entity, whose attributes include: semantic type (describing the domain specific data type); format (describing the representation of the data); collection type and collection format (used when services return a set of results); and configuration parameter (e.g., whether it is the 'main' input).

Behavioural Operations

Feta consists of four key components [Lord 05]:

(1) *Semantic Annotation.* The detailed service description is stored in the secondary publishing stage, from which Feta gathers the abstract information (the service name, the names and number of service operations, and the names and number of operation parameters) into an XML document conforming to the internal data model; Operations of manual annotation are provided which allow converting the full service description (in its original form) into the internal data model (using ^{my}Grid defined ontology). ^{my}Grid uses a suite of ontologies (expressed in DAML+OIL) to provide service classifications, and a vocabulary for expressing service descriptions;

(2) *Service Publication.* The publication operations allow services to be published in a UDDI registry; and

(3) *Query and Service Discovery.* The operations support 3 types of query [Lord 05]: a) queries of service/resource's properties described by the publisher, such as its ownership, location, or accessibility. Such a query can use UDDI technology; b) queries of opinion, observable behaviour or previous usage by third parties (users, domain experts, organisational administrators), such a query can be performed as an RDF query upon DAML-S profile; and c) queries of properties expressed using concepts from a domain specific ontology.

Feta uses an ontology on top of UDDI to deal with the semantic diversity of bioinformatics services and applications. However, many system entities of UDDI seem irrelevant and redundant, and the service annotation and discovery processes seem inefficient. In June 2009, ^{my}Grid released a new bioinformatics web service registry known as Bio-

Catalogue[9]. The major improvements of BioCatalogue include: to make use of Web2.0 technology, in particular, the tagging approach to ease the service annotation and discovery; and to implement a monitoring mechanism to report the reliability and stability of web services registered [Bhag 10]. These features are very close to those provided by the binding service proposed by this work, yet, in a specific domain application context.

3.1.4 An Ontology-based Integration Approach – BIRN

While Feta combines UDDI and ontology approaches, the Biomedical Informatics Research Network (BIRN)[10] implements a pure ontology-based integration approach. BIRN is an initiative within the National Institutes of Health that fosters large-scale collaborations in biomedical science by utilising the capabilities of the Grid infrastructure. Each participating institution maintains a database of their experimental or computationally derived data, and the data integration system performs semantic integration over the database to enable researchers to perform analyses based on larger and broader datasets [Asta 06].

Data Structure

The core component of the BIRN infrastructure is the integration engine 'mediator', which uses an ontology-based lexicon, BIRNLex, to combines multiple databases (each with a unique schema) into an accessible format or data federation. BIRNLex reuses relevant terminologies, taxonomies and ontologies when they exist, or extends them when necessary. BIRN infrastructure considers the federated biological data in a variety of representations (database, image files, simulation files, and flat text files). Bindings in BIRN created by the 'mediator', are the associations of the BIRNLex terms with brain-atlas spatial references and references corresponding to data resources.

Behavioural Operations

BIRN data resources are delivered to the end users by the 'mediator' [Elli 03], which (i) creates a database at each site, (ii) creates conceptual links to the shared ontology, (iii) situates the data in a common spatial framework, and (iv) uses the mediator to navigate and query across data sources [Asta 05].

There are many other similar projects in different domains, known as the Semantic Grid systems, e.g., caBIG[11], GEON[12], and SEEK[13]. These systems are typically built on top of the Grid infrastructure, and use ontologies as the major integration strategy [Elli 03, NCRR 06, Sala 06, Alti 05]. However, the ontology-based approach is complex

[9] BioCatalogue: http://www.mygrid.org.uk/tools/biocatalogue/
[10] BIRN: www.nbirn.net
[11] caBIG: https://cabig.nci.nih.gov/
[12] GEON: http://www.geongrid.org/
[13] SEEK: http://seek.ecoinformatics.org/

and expensive, and some implementations are beginning to practise more simple and use-friendly approaches such as using tagging, for example, as BioCatalogue and AMGA do.

3.1.5 A Tag-based Metadata Catalogue Approach – gLite AMGA

AMGA Metadata Catalogue is part of the gLite[14] software stack of the EGEE[15] project [Sant 06a]. Started as an exploratory project to study the metadata requirements of the LHC[16] experiments, AMGA has been extended to address issues from different user communities, including High Energy Physics, Biomed and Earth Observation [Sant 06b]. This motivated AMGA to build a generic metadata service interface suitable for many types of applications [Sant 06b].

Data Structure

The data structure of AMGA consists of `entries`, `attributes` and `schemas`. An `entry` is the name of the data item or resource being described. An `attribute` is a (key, value) pair with type information, which essentially is a (special form of) tag. A `schema` is a logical group of `attributes`. The AMGA `entries` are associated with one or more `schemas` and inherit the `attributes` defined in those `schemas`. `Schemas` are defined by users. They allow users to structure and organise metadata as logical groups. In this way, AMGA provides a simple metadata interface which is flexible to support dynamic schemas.

Behavioural Operations

AMGA defines operations to add and remove `entries` from a `schema`, and to list the `schemas` to which an `entry` belongs. There are AMGA operations to create and delete `schemas`, as well as to add and remove `attributes` from a `schema`. AMGA metadata catalogue service uses a relational database to store metadata. Each schema is a table, `entries` are rows and `attributes` are columns. `Attributes` are added or removed from a schema by adding or removing columns from the schema table. A master table keeps the index of all schemas. This structure means that most operations only need two accesses to the database: one to the index table and another to the table of the schema. At the back-end, AMGA supports several storage systems by way of modules [Sant 06b]. Most of storage modules provided by AMGA are for relational databases, including PostgreSQL, Oracle, MySQL and SQLLite. There is also a stand-alone implementation that supports file systems. In order to access different types of storage, AMGA queries are simply two text strings: the query statement and the name of query language being used. At the front-end AMGA supports two access protocols, SOAP and TCP Streaming.

[14] gLite – Lightweight Middleware for Grid Computing: http://glite.web.cern.ch
[15] EGEE– Enabling Grids for E-sciencE: http://public.eu-egee.org/
[16] LHC: http://www.lhc.ac.uk/

AMGA uses tags for metadata descriptions, which exposes a simple interface allowing non-technical users to use it easily [Sant 06b]. Tag metadata structure supports dynamic schemas thus can server a broad range of applications [Sant 06b]. However, for many data resources already having metadata descriptions, AMGA is unable (or inefficient) to import or link the existing metadata. The existing metadata may be used by other applications, and sometimes it is necessary to preserve their original semantics and syntax. A lack of the capability for linking external metadata makes AMGA less feasible for serving a larger number of existing data and metadata resources. The following approach resolves this problem by implementing an independent binding model.

3.1.6 An Independent Binding Service Approach – OntoGrid

OntoGrid[17] was an eight-partner EU FP6 project launched in October 2004 to investigate fundamental issues in Semantic Grids, and to integrate semantic web and Grid technologies [Gobl 05]. The Semantic-OGSA (S-OGSA) model of OntoGrid extends the Grid middleware capabilities by providing Semantic Provisioning Services and Semantically Aware Grid Services.

Data Structure

S-OGSA defines `Grid Entities` and `Knowledge Entities` (e.g., ontologies, rules, text), and bindings of the `Grid Entities` and the `Knowledge Entities` in S-OGSA, the so-called `Semantic Bindings`, are explicitly defined and independently managed. The `Semantic Bindings` are (possible temporary) metadata assertions on Grid entities and are Grid resources with their own identity, manageability features and metadata [Alpe 06].

Each Semantic Binding is associated with one lifetime state. When a Semantic Binding is created, the state is `Stable`. It will change to `GridRefsOutdated` if any related Grid Entities are updated or destroyed, or change to `ContentRefsOutdated` if the related ontologies are updated or destroyed. The Semantic Binding itself can be updated, in which case the state will remain `Stable`. When in `GridRefsOutdated` or `ContentRefsOutdated` state, the Semantic Binding will check the related Grid Entities or ontologies, if they have been updated, the Semantic Binding returns to `Stable` state, if they have been destroyed, the state will change to `Stale`. When in `Stale` state, a Semantic Binding can be requested to be `destroyed` or to be `archived`.

Behavioural Operations

The Semantic Binding Service of the S-OGSA [Corc 06] provides four basic operations: (1) `Create` (Creating a Semantic Binding of the Grid Entities and Knowledge Entities);

[17] OntoGrid: http://www.ontogrid.eu/

(2) `Add/Remove` Grid Entity References and `Add/Remove` Knowledge Entity References; (3) `Get/Update` Semantic Binding Content; and (4) `Query`, and `QueryWithInference`.

Two types of notifications have been addressed: (1) notification received by the Semantic Binding, related to the changes done to the related Grid Entities or Knowledge Entities (GridEntity/Ontoloty UpdateTime, UpdateReason will be sent); and (2) notifications from the Semantic Binding to its potential requestors when there is any change in its state or content (UpdateTime, UpdateType, NewState, UpdateReason will be sent).

S-OGSA presents an advanced binding model where bindings are independently managed. This makes it possible for S-OGSA to incrementally incorporate semantics into Grid services without disrupting existing services [Corc 07]. However, S-OGSA is designed for Grid applications and addresses special issues that add semantic descriptions to Grid service. It is not designed for general data-metadata bindings.

3.1.7 Conclusion

This section has presented existing binding management approaches used by scientific applications. Table 3.1 summarises these approaches.

Table 3.1: Summary of Related Work on Binding Management

Approach	Examples	System Features	Limitations
Simple Metadata Catalogue	MCAT	Support data intensive applications. Using logical name space for data integration.	*Dependent* binding management. Non-generic binding representation.
Web Service Registry	UDDI	Support various web services.	*Dependent* binding management. Inefficient support for scientific apps.
Semantic Web Service Registry	Feta	UDDI+ Ontology Support semantic diversity.	Inefficient annotation and look up. Irrelevant UDDI entities.
Ontology-based Integration	BIRN	Grid + Ontology Support semantic diversity.	Complex, expensive ontology modelling, implementation.
Tag-based Metadata Catalogue	AMGA	Simple, flexible metadata annotation. Support generic Grid applications.	Inefficient support for existing metadata.
Independent Binding Service	OntoGrid	*Independent* binding management.	Application dependent.

Many scientific applications require to integrate various resources. To support semantic diversity, systems such as Feta and BIRN, use ontology-based approach which is complex and expensive. Many other systems, such as BioCatalogue and AMGA, are beginning to adopt more simple and user-friendly technologies, i.e. tagging approach. Experiences from AMGA show that a tagging interface allows non-technical users to annotate data resources (or data items) more easily and efficiently.

On the other hand, lessons learned from UDDI show that in order to support scalability, a system should keep the data model as simple as possible, and avoid hierarchical or any complicated data structure.

Finally, S-OGSA provides useful experiences of implementing an independent binding management approach, and demonstrates advantages of this approach, i.e., the independent binding management allows S-OGSA to incrementally link Grid services and their semantic metadata without disrupting existing systems. However, the S-OGSA binding service is not designed for general data-metadata bindings. In contrast, the proposed binding service provides a generic approach to the independent binding management.

3.2 Data Reference Approaches

An independent binding management approach implements a data-reference model. This section introduces state-of-the-art applications of the data-reference model, including Web 2.0 Data Mashup, iRODS in the Data Grid, and Linked Data in the Semantic Web.

For easier observations, the structures of studied models have been divided into layers based on functions and locations of components. Four layers are identified: the layer of data *sources*, of *integration*, of *aggregation*, and of *presentation*. The *source* layer represents data resources typically located at provider's sites, and a reference model exposes an interface onto it for data access and query. At the *integration* layer, the different types of data pulled from a variety of data resources are converted into uniform representations which can be resolved by the reference system. The *aggregation* layer typically registers the data references and offers operations for lookup and controls. The *presentation* layer presents the information to the end users. In the following, the chosen reference models are analysed through this four-layer framework.

3.2.1 Web 2.0 Data Mashup Approach

Web2.0 Data Mashup is a content aggregation technology, which combines different types of data from different resources in a single web page. As examples, MashMyData[18] developed by Jon Blower and his team at the Reading e-Science Centre (ReSC), uses the Web Map Service[19] to provide a location, while the display of data taken from other sources of environmental data. Similarly, work led by Michael Batty at the Centre for Advanced Spatial Analysis (CASA), such as Maptube[20] and London Profiler[21], make use of the visualisation interface provided by Google Maps API, and allow geographic data from various sources to be quickly and easily overlayed.

[18] MashMyData: https://code.google.com/p/mashmydata/. (The project webpages and source codes are host at Google Code.)

[19] OGC standard – Web Map Service: http://www.opengeospatial.org/standards/wms

[20] Maptube: http://www.maptube.org/

[21] London Profiler: http://www.londonprofiler.org/

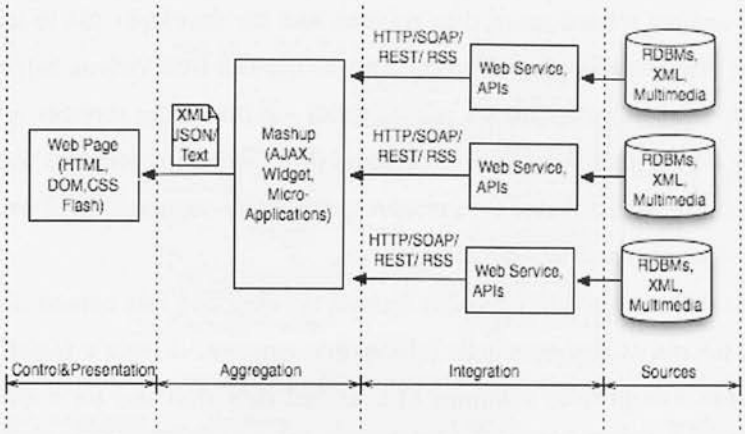


Figure 3.1: Web2.0 Data Mashup Model

Data for mashups come from numerous sources. Major web data sources, such as Google, Yahoo!, Flickr, Delicious, Facebook, Amazon and eBay, make their data accessible through web services, RSS feeds, or proprietary APIs. A mashup application typically aggregates the data from multiple sources, retrieves them at runtime, and remixes them to provide new views of the information. The website Programmableweb[22] lists ~2500 vendors' APIs and ~5500 Mashups based on these APIs[23].

Figure 3.1 depicts the structure of Web2.0 Data Mashup model. The data resource in the *source* layer can be a relational database, XML, multimedia, or programatically generated by a running application. At the *integration* layer, different types of data resources are made web-accessible through web services or proprietary APIs, which can be delivered by standard network protocols, such as HTTP, SOAP, REST, RSS, and processed by mashup applications. At the *aggregation* layer, the mashup applications typically sustain a collection of links to the data by encoding the URIs of the data within workflow functions, and using AJAX, widget, or microformat technologies to connect the resources and remix the information on-the-fly. The numbers of data connections supported by mashup systems are relatively small. For example, in 2009, Yahoo!'s Pipe[24] system supports 43 data resource connections through widgets, and MS' Popfly supports 300 widgets [Marj 09]. Finally, at the *control and presentation* layer, the information is delivered as web pages to web browsers.

Web2.0 Data Mashups do not assign globally unique identifiers to data items, which makes it impossible to set links between items in different data sources in order to connect data into a global data space. This means that recursive construction of mashups to generate further composite content is not straightforward. The implementations of a mashup

[22] Programmableweb: http://programmableweb.com
[23] At the time of writing in Dec. 2010,
[24] Yahoo!'s Pipe: http://pipes.yahoo.com/pipes

are normally against a fixed set of data sources, and the developer has to manually define attributes and integrate data sources. Due to the fact that various external sources are combined, availability becomes a crucial factor – if one of the services used is out of operation, the whole mashup can fail to function [Marj 09]. This suggests that validation of the accessibility of distributed data resources should be supported by a data-reference system, e.g., to be provided as a default function.

The Web2.0 Data Mashup approach focuses on easy and fast constructions of data visualisation interfaces that combines information aggregated from a (small) set of resources. It does not provide solutions to a unified data structure for a data-reference model, nor does it specify standard operations such models should have. By contrast, the design of the binding service defines a binding data structure and provides applicable operations. A prototype built upon this model is able to manage large scale data referenced from various resources.

3.2.2 Rule-based Data System in the Data Grid

The integrated Rule-based Data System (iRODS), a successor to SRB (see subsection 3.1.1) and held at the Renaissance Computing Institute Europa Centre (RENCI) at the University of North Carolina, supports the formation of shared collections of data in a Data Grid, while automating the execution of management policies [Moor 08b].

The architecture of the iRODS system is illustrated by Figure 3.2. At the *source* layer, distributed data resources reside in different types of storage systems, located in different institutions that are linked by the Internet. At the *integration* layer, different types of data resources are managed by sets of iRODS micro-services which are atomic or combined standard operations that are invoked through user-defined and administrator defined rules. At the *aggregation* layer, the federation of servers is implemented with a central metadata catalogue that manages persistent state information, and a rule engine that applies rules stored in a rule base. For each desired output, rules are defined that control the execution of the remote operations. At the *control and presentation* layer, clients issue requests to the iRODS systems to invoke rules.

Data in iRODS are typically in file formats. Each file is assigned a unique Global File Identifier (GFI), which is the name of collection and data object as they appear in iRODS. The metadata catalog manages the mappings of the GFIs to the location of data files, the mapping of the GFIs to the location of metadata, and the association of the data to the metadata through GFIs. iRODS federates user-defined rules and constraints that are used to manage state transitions within a data grid. Rules can be published for use by others. By default, iRODS provides three types of rules, including user-access rules which allow users to view and manipulate the collections and files; procedural rules used

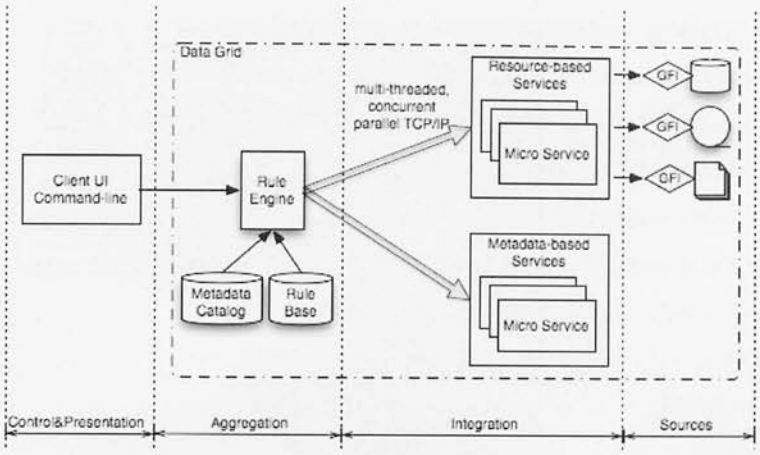


Figure 3.2: iRODS System Model

for data access and transfer (this does not includes content transformation for integration purposes); and consistency rules used for verifying consistency states of data and system.

Some known issues of iRODS include how to support the application of dynamic rules and constraints on collections of large scale data; how to avoid deadlocks of rules and constraints; how to minimise complexity for handling integrative rules; and how to keep the size of the rule space manageable [Moor 08b].

iRODS references data from storage systems installed on Data Grids. iRODS has control over the storage systems and can execute the rules at each storage location where data resides. By contrast, the proposed binding service addresses the challenging issues that data resources are located at external storage, and binding service may not have full control over the storage systems.

3.2.3 Linked Data Approach in the Semantic Web

The recently emerged term Linked Data refers to a set of best practices for exposing, sharing and connecting pieces of data, information, and knowledge on the Semantic Web using URIs and RDF [Bize 09b]. Tim Berners-Lee outlined a set of principles for the Linked Data [Bern 06]:

1. Use URIs as names for things;
2. Use http URIs so that people can look up those names;
3. When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL); and
4. Include links to other URIs, so that they can discover more things.

With the efforts of W3C Linking Open Data (LOD) community, an increasing number of data providers have started to publish and interlink data on the Web. Currently[25],

[25] In Dec. 2010

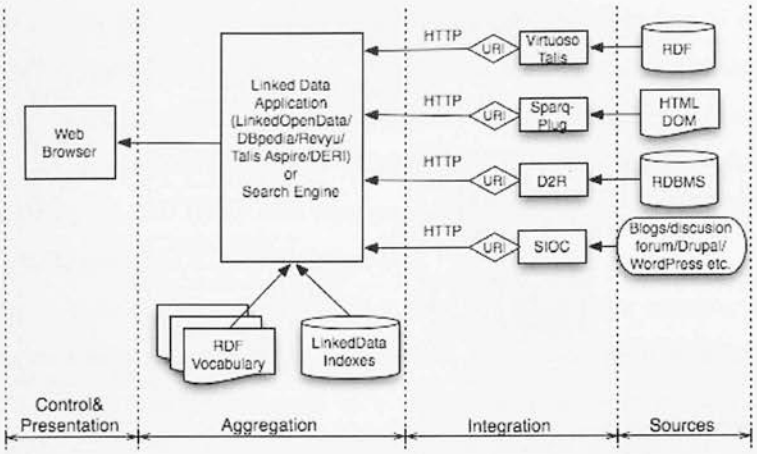


Figure 3.3: The Linked Data Model

the Semantic Web consists of several billion RDF triples and covers domains such as geographic information, people, companies, online communities, films, music, books and scientific publications[26].

Figure 3.3 shows the structure of Linked Data application model. The data *sources* are typically RDF files or RDF triple stores. Other types of resources, such as databases or documents in HTML, are also allowed. RDF is used to make typed statements that link arbitrary things in the world. Things are represented as URIs. It is possible to use different URIs to identify the same entity, so called 'URI aliases'. Vocabularies are used to describe entities and how they are related. Some well-know vocabularies include, FOAF[27], SIOC[28], SKOS[29], DOAP[30], vCard[31], Dublin Core[32], OAI-ORE[33] and GoodRelations[34].

At the *integration* layer, various tools are made available for mapping different types of resources to Linked Data, or rewriting query URIs into SPARQL for access. For example, D2R [Bize 06] maps relational database to Linked Data, SparqPlug [Coet 08] extracts Linked Data from HTML documents, SIOC Exporters[35] provide the Linked Data wrappers for blogging engines, content management systems and discussion forums.

At the *aggregation* layer, a typical Linked Data application aggregates the Linked Data and generates useful information. For example, Tabulator[36] and Marbles[37] track the

[26] http://linkeddata.org/ Retrieved Dec. 2009
[27] The Friend of a Friend (FOAF) vocabulary: http://xmlns.com/foaf/spec/
[28] Semantically-Interlinked Online Communities (SIOC) ontology: http://sioc-project.org/ontology
[29] Simple Knowledge Organization System (SKOS): http://www.w3.org/2004/02/skos/
[30] Description Of A Project (DOAP): http://trac.usefulinc.com/doap
[31] vCard -The Electronic Business Card: http://www.imc.org/pdi/vcard-21.txt
[32] Dublin Core: http://dublincore.org/
[33] Open Archives Initiative Object Reuse and Exchange (OAI-ORE): http://www.openarchives.org/ore/
[34] GoodRelations-The Web Ontology for E-Commerce: http://www.heppnetz.de/projects/goodrelations/
[35] SIOC Exporters: http://sioc-project.org/wordpress
[36] Tabulator: http://www.w3.org/2005/ajar/tab
[37] Marbles: http://marbles.sourceforge.net/

provenance of data while merging data about the same thing from different sources. Tabulator also allows users to discover and highlight a pattern of interest and query for similar patterns in the data web. Linked Data search engines, such as: Falcons[38], SWSE[39], Swoogle[40], Sindice[41], and Watson[42], crawl and index Linked Data from the web by following RDF links, and provide query capabilities over the aggregated data.

At the *presentation* layer, applications such as Linked Data browsers allow users to navigate the sources link by link. Advance Linked Data browsers combine mashup technology to provide useful functions, for example, DBpedia Mobile [Beck 08] can, based on the current GPS position of the mobile device, provide a location-centric mashup of nearby locations from DBpedia[43], photos via a Linked Data wrapper around the Flickr API, and associated reviews from Revyu[44].

The Linked Data are accessed through SPARQL queries. SPARQL is a data-oriented query language that queries information held in the RDF data model or RDF graph. A solution to the query is obtained by matching graph patterns to subgraphs of the target RDF graphs [Anto 09, Guti 09]. The W3C SPARQL specification[45] defines four query forms: CONSTRUCT, DESCRIBE, SELECT, and ASK. CONSTRUCT and DESCRIBE return an RDF graph as query result; SELECT returns an RDF graph constructed by substituting variable in query patterns; and ASK returns an RDF graph that describes the resources found. The resulting RDF graph can be presented by using the RDF/XML or N3 format [Anto 09, Guti 09]. SPARQL provides only retrieval functionality without capabilities of update (creation, update, deletion). SPARQL does not support reasoning-based semantic matching, i.e., it does no support RDF schema based reasoning [Anto 09, Guti 09].

Standard service-based access mechanisms for RDF data have been taken up by some community organisations, such as the Database Access and Integration Service Working Group (DAIS-WG) in the Open Grid Forum (OGF). The DAIS-WG specification, WS-DAI, provides advanced RDF(S) query facilities, including the support of distributed RDF queries. WS-DAI allows indirect access to RDF(S) data, which creates intermediate results and delivers them to the distributed query process [Guti 09].

The performance of SPARQL is less efficient than SQL language [Bize 09c, Bize 08]. Comprehensive measures and empirical results have been provided by previous research, such as the Berlin SPARQL Benchmark[46], which shows SQL is significant faster than

[38] Falcons: http://iws.seu.edu.cn/services/falcons/objectsearch/index.jsp
[39] SWSE: http://swse.deri.org/
[40] Swoogle: http://swoogle.umbc.edu/
[41] Sindice: http://sindice.com/
[42] Watson: http://watson.kmi.open.ac.uk/WatsonWUI/
[43] DBpedia: http://en.wikipedia.org/wiki/DBpedia
[44] Revyu: http://revyu.com/
[45] SPARQL Query Language for RDF: http://www.w3.org/TR/rdf-sparql-query/
[46] Berlin SPARQL Benchmark: http://www4.wiwiss.fu-berlin.de/bizer/BerlinSPARQLBenchmark/

SPARQL in query (data loading, and supporting multiple clients) [Bize 09c]. Moreover, in most cases, the query statements of SPARQL are more complex than SQL and harder to interpret [Bize 09c, Bize 08].

Besides SPARQL-query issues, research challenges for Linked Data include but are not limited to: how to answer expressive queries against the Web of Data by on-the-fly link traversal or federated querying in a timely fashion [Bize 09b]; how to detect the dead links when the objects are no longer maintained and detect potential links when new entities are being added [Volz 09, Hass 09, Auer 09]; how to integrating multiple data items representing the same real-world object into a single, consistent, and clean representation [Blei 08]; how to ensure the data most relevant or appropriate to a user's needs are identified and made available [Bize 09a]; how to represent the provenance and trustworthiness of data drawn from many sources into an integrated view [Bern 06].

The Linked Data model addresses many similar issues to the binding service. But the two designs are different in various aspects, such as data structure, binding representation, storage and search mechanisms. In Chapter 8, detailed comparisons of the two approaches will be presented.

3.2.4 Conclusion

Table 3.2 summaries the data-reference approaches presented in this section.

Table 3.2: Summary of Related Work on Data Reference Approach

Method	Examples	System Features	Limitations
Web2.0 Data Mashup	MashMyData, Maptube	Aggregate distributed web data. Runtime data retrieval. Provide new views of information.	Not use globally unique IDs. Inefficient aggregation process. Not support resource validation.
Rule-base Data System	iRODS	Data are global unique identifiable. Sharable aggregated information. Support data intensive applications.	Complex, non-scalable rule space. Difficult to maintain consistency.
Linked Data	DBpedia, Tabulator	Support sharing, reuse various web data which are identifiable by URIs.	Complex RDF modelling. Insufficient query support.

Web2.0 Data Mashups focus on easy and fast constructions of data visualisation interfaces, and support runtime data retrieval from various resources. However, the implementation of a mashup is often against a fixed set of data resources, and provides only basic data-reference behaviours, for example, normally there is no validation mechanisms to check the accessibility of distributed data resources. iRODS is an advanced Data Grid system which supports automatic executions of user-defined constraint rules and uses micro-services to reference data from distributed storage systems. Unlike the binding service, iRODS does not address the issues that storage systems are beyond the full control of the iRODS system. The Linked Data approach addresses many similar issues to the bind-

ing service. However, the binding service implements a different approach. The design of the binding service will be provided in Chapter 5. After that, a detailed comparison of the Linked Data model and the binding service model will be presented in Chapter 8.

3.3 Related Technologies

This section introduces technologies used in design or building the binding system, including tagging and Cloud technology.

3.3.1 Tagging Technology

In the above examples, tagging has been noted as an efficient annotation strategy. The proposed binding system adopts the tagging approach for binding annotations. This subsection reviews existing research efforts in the area.

Although, the use of free-form labelling through tagging may appear like a recipe for disaster, surprisingly it has turned into one of the significant trends on the web [Chi 08]. The success of Web2.0 tagging systems, Delicious[47] – a web bookmark system, Flickr[48] – a web photo system, YouTube[49] – a web video system, and CiteULike[50] – an academic paper citation system, has confirmed the usability of tags in annotation, cataloguing, search and scalable processing. Tagging provides an effective and flexible annotation and search mechanism which is suitable for large and heterogeneous data discovery in a distributed computing environment.

Tags may be ambiguous, and are often incorrectly applied. However, being a power law phenomenon [Dell 08, Halp 07], over time, tags can slowly stabilise to a pattern [Gold 05, Halp 07], and a loose categorisation system for resources will emerge. Vocabulary that organically grows from a diverse set of users has been described as a 'folksonomy' [Wal 07, Voss 07]. Golder and Humberman observe the data resources of delicious, and show such stabilisation patterns in which the proportion of each tag is a fixed percentage of the total frequency of all tags used. The phenomenon has been interpreted by the stochastic Polya urn model, which can be illuminated by a simulation using an urn and balls. In its simplest form, the stochastic probabilistic model consists of an urn initially containing two balls, a red ball and a black ball. In each step of the simulation, a ball is randomly selected from the urn and replaced in the urn along with an additional ball of the same colour. After a large number of draws the fraction of the balls with the same colour stabilises, though the fractions converge to random limits in each run of the simulation [Gold 05].

[47] Delicious: www.delicious.com
[48] Flickr: http://www.flickr.com/
[49] YouTube: youtube.com
[50] CiteULike: CiteULike.org

Golder and Humberman also indicate the stabilisation might be due to the imitation behaviour of users in that they tend to select tags that have been used by other users [Gold 05]. The principle of 'social proof' suggests that actions are viewed as correct to the extent that one sees others doing them [Cial 01].

Tagging systems provide an 'intellectual ramp' [Atki 09] in that a new user can start by adding a small number of freely chosen tags without any need to study the correct choice of tags or the correct format. They can then incrementally learn about and adopt emerging conventions (from 'folksonomy') or use formally agreed tags adopted by their community or topic specialists. They can also easily revisit and add tags as they have the need.

For different tagging systems, the internal tag data structure can be different, i.e., in the form of key-value pair, a list, a block or a reference. When necessary, semantic and syntax constraints can be included, i.e., it is possible to use ontology URIs as the tags, thus achieving the precision of interpretation of semantics, but the freedom of construction of tagging. It is also possible to provide user-friendly front-end which allows users to select tags from the tag set defined.

To initiate content retrieval, social tags are often shown in a *tag-cloud*, a visual depiction of tags in which the most popular tags are typeset in a larger font or more prominent colour [Clem 10]. The relevance of a tag is often based on the global popularity of the tags in the entire network. The *tag-cloud* provides a way of navigation where the query only uses a single popular word as the query term, resulting in many retrieved objects [Clem 10].

Besides the *tag-cloud*, most tag-based systems provide query interfaces which normally apply traditional information retrieval (and web search engine) approaches, and use the selected tags as query term and rank the retrieval content according to popularity or freshness [Clem 10]. The query interface usually allows users using multiple-word queries to disambiguate their information needs. The relationship among the input query tags can be either intersection or union. For example, the query variables supported by Wordpress[51], a web content software, include: `tag_add` which retrieves web contents that have ALL of the specified tags; `tag_in` which retrieves the web contents having ANY one of the specified tags; `tag_not_in`, which retrieves the web contents having NONE of the tags specified.

The proposed binding service adopts the tagging approach to provide a simple and efficient binding annotation interface. A binding tagging framework will be described in Chapter 5 (section 5.1).

[51] Wordpress: http://wordpress.org/

3.3.2 Cloud Technology

The Cloud technology is used as underpinning architecture to provide scalability for the binding system. This subsection introduces this technology.

Architecture evolves overtime. In the 1960s and 1970s, the first wave of computing consisted of large expensive, labour-intensive, monolithic servers. Internal resources were pooled and heavy use was made of virtualisation to ensure the best use of these very expensive resources [Benn 09], yet, such virtualisation was at lower level than compared with now. In the 1980s and 1990s, with the rise of PCs and the shrinking costs of networking and computing infrastructure, the client/server structure became popular which provided the ability to split the application tier away from the server tier. This has efficiently supported distributed clients running richer user interfaces and also reduced costs by offloading the (user handling) application workloads from monolithic servers [Benn 09]. In the 2000s, as datacentres[52] started to fill up and power, space and cooling became more and more expensive, concepts such as Grid Computing began to be established to enable services and resources sharing among virtual organisations [Benn 09]. And in recent years, Cloud Computing takes these concepts further by offering much more simplicity in use and deployment, and more automated dynamic resource and workload management practices [Gros 08, Benn 09].

In a Cloud, data and computing are co-located at a centralised server farm or a datacenter and accessed as well-defined services [Szal 09].

When a Cloud is made available in a pay-as-you-go manner to the general public, it is so-called as a *Public Cloud* [Armb 09]; the service being sold is *Utility Computing*. It can be divided into three levels of service offering: Service-as-a-Service (*SaaS*) which delivers applications as service, Platform-as-a-Service (*PaaS*) which delivers application development and deployment platform as service; and Infrastructure-as-a-Service (*IaaS*) which delivers hardware (service, storage, and network) and associated software (operating systems virtualisation technology, file system) as a service. Examples of *Public Cloud* include, Amazon Elastic Compute Cloud (EC2)[53] and Simple Storage System (S3)[54], Flexiscale[55], AppNexus[56], NewServers[57], JungleDisk[58], and ElasticHosts[59]. Research challenges in this area include utility computing and economic models [Armb 09, Buyy 08],

[52] A datacentre or data centre, (also called a server farm), is known as a facility used to house computer systems and associated components.

[53] Amazon Elastic Compute Cloud (EC2): http://aws.amazon.com/ec2/

[54] Amazon Simple Storage System (S3): http://aws.amazon.com/s3/

[55] Flexiscale: http://www.flexiscale.com/

[56] AppNexus: http://www.appnexus.com

[57] NewServers: http://www.newservers.com

[58] JungleDisk: http://www.jungledisk.com/

[59] ElasticHosts: http://elastichosts.com

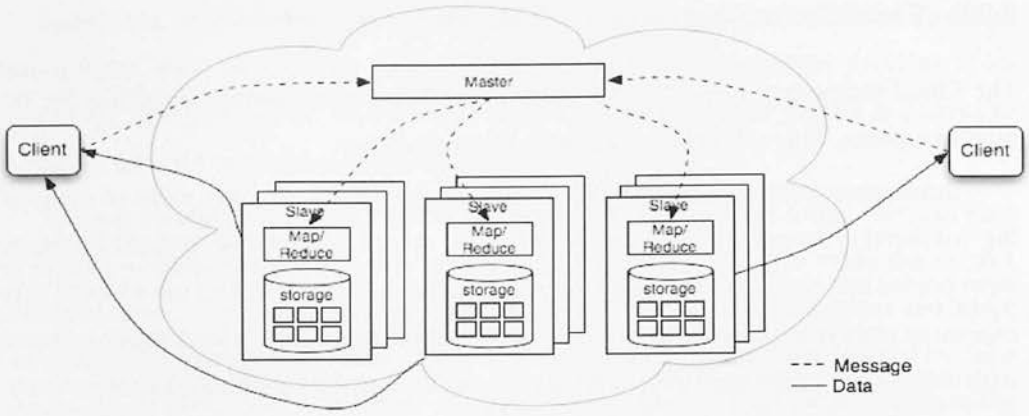


Figure 3.4: A Cloud Model

interoperable interaction and virtualisation [Harm 09], platform computing and dynamic provisioning [Benn 09, Blan 09].

A *Private Cloud* refers to internal datacenters of an orgnisation, not made available to the general public. Examples of Private Cloud include GrayWulf [Szal 00b, Szal 09], OpenCirrus [Camp 09], Google File System [Ghem 03], and BigTable [Chan 06]. Technologies for building Private Cloud include Hadoop[60], HBase[61], Hypertable[62], Eucalyptus[63], and Sector[64]. Active researches in this area are around the topics of: Cloud database and data partitioning such as BigTable [Chan 06], GrayWulf [Szal 00a], Cassandra [Laks 08], Sector [Gu 09], and Gamma [DeWi 90]; parallel programming languages such as Google's MapReduce programming Model [Dean 04], Yahoo!'s Pig Latin [Olst 08], and Sphere [Gu 09]; and Cloud testBed [Gros 09, Nurm 09]. The issues of the *Private Cloud* are of interest.

A typical Cloud model is depicted in Figure 3.4. A Cloud normally consists of a master node and a set of slave nodes[65]. A master node keeps metadata of storage and files, and is responsible for coordination and management of the slave nodes, while the slave nodes provide the actual data storage and data processing. Each slave node normally comprises a data store and a Map/Reduce processing layer. Data in the slave nodes are normally replicated for failure recovery. Client requests are received by the master node which further sends Map/Reduce tasks to the slave nodes. The slave nodes involved execute the tasks in parallel, and deliver the results to the client directly.

[60] Hadoop: http://hadoop.apache.org/

[61] HBase: http://hadoop.apache.org/hbase/

[62] Hypertable: http://www.hypertable.org/

[63] Eucalyptus: http://www.eucalyptus.com/

[64] Sector: http://sector.sourceforge.net/

[65] Here, a simple Cloud model is discussed. Embedding one master in the cluster potentially introduces a single-point-of-failure. In practice, there can be multiple master nodes.

Map/Reduce is a programming model originated from Functional Programming paradigm. It is used by some Cloud frameworks, such as Google File system, and Hadoop, for users to write distributed applications.

Map/Reduce expresses a large distributed computation as a sequence of distributed operations on datasets of key/value pairs. In the map phase, the framework splits the input datasets, a set of key/value pairs, into a large number of fragments and assigns each fragment to a map task, and all the map tasks are then distributed across the cluster of nodes on which it operates. For each input key/value pair (K, V), the map task invokes a user defined map function that processes the input and produces a result key/value pair (K', V'). Following the map phase the system sorts the intermediate datasets by key and produces a set of (K', V'*) tuples so that all the values associated with a particular key are grouped together. In the reduce phase, each reduce task processes the fragment of (K',V'*) tuples assigned to it. For each tuple it invokes a user-defined reduce function that transfers the tuple into an output key/value pair (K'', V'').

Programs written in this functional style are automatically parallelised and executed on a large cluster of commodity machines [Ghem 03].

In this thesis, one of the Cloud technologies, Hadoop, is used to provide scalability properties for the binding service. The experiments with a Hadoop-based binding storage will be presented in Chapter 6 (section 6.2).

3.3.3 Conclusion

This section has introduced tagging and Cloud approach. A summary is provided in Table 3.3.

Table 3.3: Summary of Related Technologies

Method	Examples	System Features	Challenges
Tagging	Flickr, Delicious	Support flexible, use-friendly annotation and large heterogeneous data discovery.	Tags are noisy, ambiguous.
Cloud Cluster Map/Reduce	GFS, Hadoop	Scalable, elastic utility, cheaper. Programmable interface. Support parallelisation.	Data and computing partitioning. Utility vs. Cost. Interoperability.

Experiences from both Web2.0 (e.g., Flickr, Delicious) applications and Grid systems (e.g., AMGA, BioCatalogue) have shown that tagging is an effective and flexible annotation and search mechanism which is suitable for large and heterogeneous data discovery in a distributed computing environment. While Cloud technology recently emerges as a possible solution to scale up a system architecture. A later chapter will describe how the binding service makes use of these technologies to provide advanced system features.

3.4 Summary

This chapter has explored prior work in different areas, including: binding management, data-reference models, tagging approaches and Cloud Computing. Recall a previous chapter (in section 2.4) has summarised the typical binding scenarios by giving 6 sketched pictures. Considering the target binding scenarios and applying the experiences learned in literature, it makes sense to draw out the following principles for the design of binding systems:

- A binding system should provide simple and efficient annotation APIs as well as support semantic diversity; (Applies to **Binding Scenario 1**.)
- It should be able to support data intensive applications, and handle large scale bindings; (Applies to **Binding Scenarios 2, 3, 5, 6, 7**.)
- It should provide integration mechanisms allowing easy and efficient aggregation and query of various distributed data and metadata; (Applies to **Binding Scenarios 2, 3, 5, 6, 7**.)
- It should provide a generic binding representation and an independent service interface, which can support a wide range of scientific applications; (Applies to **Binding Scenarios 1, 2, 3, 4, 5, 6, 7**.)
- It should allow the aggregated information to be easily and efficiently shared and reused. (Applies to **Binding Scenarios 4, 5, 6, 7**.)

In Chapter 5, a novel approach to independent binding management is introduced that combines data-reference and tagging approaches, which can be widely applicable to various e-Science applications. The next chapter discusses the formal foundation of the binding systems.

Chapter 4

A Formal Binding Model

To improve the clarity of the concepts, this chapter develops a formal binding model. The chapter is organised as follows: Section 4.1 gives a brief introduction of background knowledge and motivation. Section 4.2 describes the formal binding model consisting of binding structure and behavioural operations. Section 4.3 defines a consistent binding model and discuss how to deal with binding failures. Section 4.4 discusses possible usage of this approach. Section 4.4 summarises this chapter.

4.1 Introduction

In Chapter 2, an informal description of bindings has been given to provide a way to talk about bindings. As shown in Chapter 3, various approaches to metadata and binding management exist. As yet, there is no formal foundation for the model concepts that would clarify the problems related to the use of binding in a scientific data and metadata context.

This chapter offers a first attempt at a formal binding model which captures binding characteristics and management functions. A binding-oriented approach is proposed, where bindings are first-class citizens and have an existence independent of their associated data and metadata. The binding model consists of formal structures for representing bindings and of behavioural operations on those structures. The intention is to present the overall shape of the binding model in just enough detail to clarify and explore the computational properties of such a model, and further justify the thesis that to establish a general-purpose binding management framework is a worthwhile and feasible undertaking. We focus on the issues related to *independent* binding management, however the fundamental principles are applicable to *dependent* binding management.

Existing approaches to relationship modelling include: (1) the *attribute* approach, such as Ontology and RDF models [Grub 93, Tamm 02, W3C 04], and UML models [Kim 02, Evan 98], which model a relationship as properties or attributes of an object

or entity and focus on the static characteristics of that relationship; and (2) the *function* approach [Fan 09b, Fan 09a, Boha 06] which view relations as (partial) functions mapping from one domain to another. Their focus is the dynamic aspect of relationships. The proposed approach in this dissertation takes a different view from the traditional approaches: rather than view bindings as attributes of objects or functions, the binding-oriented approach treats the binding relationships as first-class citizens, and models both structural and behavioural characteristics of bindings.

At the abstract level, the binding-oriented approach considers the bindings as *objects* in the model. An object-oriented data model is therefore the natural platform on which to build the binding model. A substantial amount of theoretical research has been carried out in the area of object-oriented programming languages and databases [Atki 89, Rhod 01, Voss 95, Lecl 88, Bert 96, Bern 00], and theoretical foundations have been established. This work extends prior research endeavours to address binding issues.

4.2 A Formal Binding Model

Consider the computational context shown in Figure 4.1. A binding service maintains persistent state of the set B, but it has no responsibility to any data in the set D or any metadata in the set M.

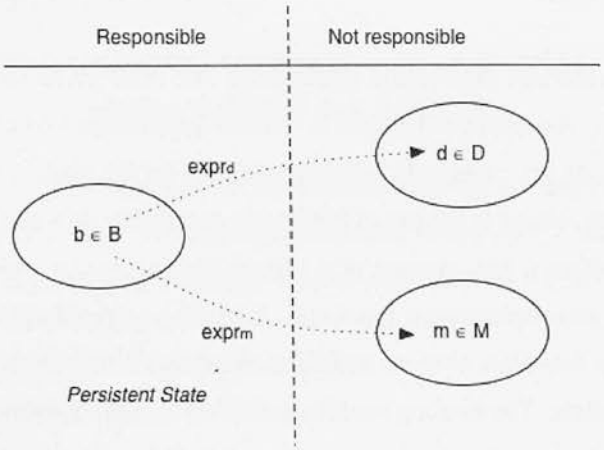


Figure 4.1: Binding Computational Context

Suppose there is a time-varying set B for each binding service. The Object-Oriented model captures the hidden state that represents the set B. Any realisations of a binding service will have some means of identifying that service, e.g., a URI or a WS-Addressing. That implementation identity corresponds to an identity of an instance of the object class introduced as part of the formal model. All of the operations supported by such a binding service implementation have to correspond with the methods of the object class introduced

as a formal model below. In summary, the Object-Oriented model is introduced to convey the persistence of the stored bindings as state in the class's instances.

Object-orientation as a paradigm is based on five fundamental principles [Voss 95, Lecl 88, Atki 89]:

1. Each entity of the real world is modelled as an object which has an existence of its own, manifested in terms of a unique identifier distinct from its value;
2. Each object is encapsulated into a structure and a behaviour, the former is described in terms of attributes with attribute values, which together represent the state of the object;
3. The state of an object can be accessed or modified by invoking corresponding methods;
4. Objects sharing the same structure and behaviour are grouped into classes, an object can belong to one or more different classes;
5. A subclass inherits the structure and behaviours (attributes and methods) from its super-classes.

Within the Object-oriented modelling framework, structural aspects are distinguished from behavioural aspects, and binding model is given as a pair of the form:

$$\mathcal{B} = (B_{struc}, B_{behav})$$

where sub-model B_{struc} captures the data characteristic of bindings, and B_{behav} defines the behavioural operations on the model structure. In the following, each component is discussed in turn.

4.2.1 Model Structure, B_{struc}

Formal definitions of object model structure have been given in previous research [Rhod 01, Voss 95, Lecl 88, Bert 96]. We revise those for the case of binding.

Given:

- A set ID of symbols called identifiers.
- A set D of data objects.
- A set M of metadata objects.
- A set A of attributes.

The definition of the binding data model is given as follows:

Definition 4.1. *Binding data model,* B_{struc}

B_{struc} is a set of objects, where

(i) Each object in B_{struc} (called a binding b) has an identifier id, $id \in ID$;

(ii) Each b has a relationship attribute, d, which represents the object o_d, $o_d \in D$, and is denoted by an expression $expr_d$ over the objects of D;

(iii) Similarly, each b has a second relationship attribute m, which represents the object o_m, $o_m \in M$, and is denoted by an expression $expr_m$ over the objects of M;

(iv) Each b may have other attributes $a \in A$.

□

A binding, b, is a four-tuple:

$$< id, d, m, \{a_0, a_1, \ldots, a_p\} >$$

with the elements defined as:

1) The binding identifier, id;
2) The relationship attribute, d;
3) The relationship attribute, m; and
4) The attributes set, $\{a_0, a_1, \ldots, a_p\}$.

Later, we use dot notation, e.g., $b.d$, and $b.m$, to express the binding attributes.

In general the binding service is independent from the services which handle the data and metadata. A requirement from some applications was that annotated relationships could be constructed, maintained and used even when the scientist had no direct access to the data or metadata. Therefore, we discriminate between $expr_d$ hold in the binding store and d, which it references, hold in some other store. The same discrimination applies to metadata (i.e., $expr_m$ and m).

The expression of d and m is given by $expr_d$ and $expr_m$. $expr_d$ and $expr_m$ are placeholders for specifying semantic and structure details of bindings to reference the data and metadata, respectively. The model does not specify their representations. They can be mutable identities or complex structures that represent the data and metadata structures. Finding appropriate representations of $expr_d$ and $expr_m$ that trade off the expressiveness of semantics with interoperability of highly diverse scientific data and metadata remains as a challenge for binding system design. (The design of the binding service addresses this issue. A plausible binding representation will be described in Chapter 5, subsection 5.1.3).

The relationship attributes d or m may be empty which is denoted as *nil*.

The *multiplicity* of the relationship between data and metadata objects d and m denoted by bindings can be *0:0*, *m:0*, *0:n*, and *m:n*, where $m \geq 1$, and $n \geq 1$.

A binding b is *directional* since data and metadata are differentiated.

Example 4.1. Consider the EurExpress system. Suppose binding b_i is created to represent the association of the gene expression images in the Set D_g with the image metadata in the Set M_e. Thus,

$b_i = < id_i$, *'/eurexpress/images/image$_j$'*, *'SELECT image_metadata FROM image_metadata_table WHERE image_Id = j'*, {*'binding_creation_date:01-Jan-2011'*, *'binding_author:Alison'*, ...}>

Here the expression for $b_i.d$ is specified as the full path of a biomedical image file, and the expression for $b_i.m$ is specified as an SQL statement. The binding attributes are used to accommodate the annotations of the image.

□

Example 4.2. Still considering the EurExpress system. The biomedical images and their metadata can be referenced by URIs in order to allow web-access. Thus, a different design would specify a EurExpress binding b_i as follows:

$b_i = < id_i$, *http://www.eurexpress.org/images/image$_j$*, *http://www.eurexpress.org/query?SELECT image_metadata FROM image_metadata_table WHERE image_Id = j*, {*'binding_creation_date:01-Jan-2011'*, *'binding_author:Alison'*, ...}>

In this case, the expression for $b_i.d$ is specified as the URI of an image file, and the expression for $b_i.m$ is also a URI which consists of an SQL statement. The binding attributes remain the same.

□

The formal model intends to capture more general binding requirements obtained from the real-world observations. For example, when $expr_d$ and/or $expr_m$ are local copies of objects, the binding data structure can be used to represent a *dependent* binding.

Example 4.3. Consider the NASA archive system PDS, where bindings are *dependently* managed. A PDS binding b_i can be created as follows:

$b_i = < id_i$, *<?xml version="1.0"?><node>...</node>*, *http://pds.nasa.gov/mars_exploration_1965* >

In this case, the expression of $b_i.d$ is an XML file, and the expression of $b_i.m$ is the reference to the information of the first mission to explore Mars. The binding b_i has an empty attribute set.

□

Depending on concrete computational contexts, binding attributes can be used to describe various binding related information. Example 4.1 has illustrated how binding attributes were used to describe biomedical images. The next example discusses a different use case.

Example 4.4. Consider a provenance application. Suppose bindings are created to represent the associations of workflow processes and their output data. Binding attributes could be used to store provenance annotations, i.e., {'user=Alison' (who), 'materials=embryo029' (what), 'experiment=ISH' (how), 'purpose=Identify gene NRG1' (why).}

□

There may be more than one binding between a given pair of data and metadata objects $<o_d, o_m>$. And there is a need to distinguish the identical bindings and equal bindings. In the following, some technical notations are to be used. If $b = <id, d, m, \{a_0, a_1, \ldots, a_p\}>$, $ident(b)$ denotes the binding identifier id, so that $ident$: $B \rightarrow ID$. $cont(b)$ denotes the binding content $<d, m, \{a_0, a_1, \ldots, a_p\}>$.

Definition 4.2. *Binding Equalities.*

(i) Two bindings b and b' are *identical* iff $ident(b) = ident(b')$.

(ii) Two bindings b and b' are *(shallow-)equal* iff $cont(b) = cont(b')$ (in the sense of mathematical pair equality).

(iii) Two bindings b and b' are *deep-equal* iff $span(b) = span(b')$, where $span(b)$ is obtained from b by replacing any expression or identifier i (in $cont(b)$) by the value of the object identified by i .

□

The definition of binding equality corresponds to objects *identity*, *shallow-equality* and *deep-equality* discussed by previous theoretical research [Lecl 88, Bert 96, Bern 00]. Intuitively, *shallow-equality* considers the equality of all the direct attributes of a binding, and *deep-equality* considers in addition to the equality of the attributes, the equality of the attributes of binding which are reachable by means of references. The rest of the work mainly discusses *shallow-equality*[1].

B, D, M may be disjoint or may overlap. If B, D, M are disjoint sets, we call the bindings in B *simple-bindings*. When B, D, M overlap, it results in *complex-bindings*,

[1]Managing the *deep-equality* can be very difficult in a heterogeneous distributed context. It has been studied in many research works, but the problem remains open. Verifying the *deep-equality* is particularly hard in the presence of the binding model defined, where data and metadata are managed independently at third-party's sites, which are beyond the control of a binding system.

including (1) the *nested-bindings*, i.e. $b.d = b'$, that is the binding b referring to another binding b' through d; and (2) the *self-referencing* bindings, i.e. $b_1.d = b_2, \ldots, b_n.d = b_1$, that is a binding contains cycles that is, it directly or indirectly makes references to itself. A *simple-binding* contains no cycles.

The *complex-bindings* are more expressive than the *simple-bindings*, i.e., in many ontology constructions the logical levels are neither flat nor hierarchical but circular [Fern 08], which needs the *complex-bindings* for descriptions. However, in practice, *complex-bindings* can increase the complexity of implementations. The choice is left to be made in the design process. (A discussion of choosing data structure for the binding service is presented in subsection 5.1.3)

4.2.2 Model Behaviours, B_{behav}

The behavioural component of the binding model, B_{behav}, provides a finite set of operations attached to the B_{struc}. A variety of operations can be provided. Firstly, four basic operations are introduced: *create*, *delete*, *update* and *query*.

For convenience, let B denote a binding-set object that supports operations on a set of bindings, an instance of B_{struc} also is denoted by B (with clarification if this could be ambiguous). Thus b is of the set B and b is a binding object managed by the binding-set object B.

The binding *create* operation is specified as follows:

Definition 4.3. *Binding create operation*
For some binding content $cont = < d, m, \{a_0, a_1, \ldots, a_p\}>$, binding creation operation on B is given by $B.create(cont)$ which generates a unique binding identifier, id, and creates a new binding $b = < id, d, m, \{ a_0, a_1, \ldots, a_p\} >$. The result of $B.create(cont)$ is $B' = B \cup \{b\}$, thus $B \subset B'$ holds.

□

Similarly, the definition of *delete* operation is given as:

Definition 4.4. *Binding delete operation.*
For some binding b, $b \in B$, binding *delete* operation on B is given by $B.delete(b)$, which removes b from B. The result of $B.delete(b)$ is $B' = B \setminus \{b\}$, thus, $B' \subset B$ holds.

□

The *update* operation is applied to a single binding object rather than the binding set B, and is defined as follows:

Definition 4.5. *Binding update operation.*
For binding set $B = B_1 \cup \{b\}$, where $b = <id, d, m, \{a_0, a_1, \ldots, a_p\}>$, and some binding content $cont = <d', m', \{a_0', a_1', \ldots, a_p'\}>$, binding *update* operation applied to b is given by $b.update(cont)$ which replaces b with $b' = <id, d', m', \{a_0', a_1', \ldots, a_p'\}>$, where $ident(b) = ident(b')$. The result of $b.update(cont)$ is $B' = B_1 \cup \{b'\}$, thus $b \in B$ and $b' \in B'$ hold.

□

The querying of a binding model is an operation on the binding model without changing its state. Intuitively, given a binding data set B, a *query* operation returns a binding data set consisting of the subset of B that conforms to a given qualification, c. Query also functions as a kind of pattern matching operation, since qualification amounts to patterns that select a subset of B. Before giving the *query* operation, we need three comparison operators that can be used in queries: ==, = and ≈.

- == operator tests for binding identity equality, that is, $b == c$ evaluates to true when b and c are *identical* (see Definition 4.2).
- = operator tests for binding value equality, that is, $b = c$ evaluates to true when b and c are *(shallow-)equal* (see Definition 4.2).
- ≈ operator tests for binding value similarity. Similarity comparison is useful when we need to look for approximate results. The operator can be any similarity metric i.e. q-grams, Levenshtein, Jaro distance, such that $b \approx c$ evaluates to true if b and c are "close" enough *w.r.t.* a specified threshold.

Definition 4.6. *Binding query operation.*
A *query* operation on B is the form: $B.query(f) = \{b \mid b \in B, \text{ and } f(b) \text{ hold } \}$, where f is a first-order Boolean predicate, taking the value *true* if $[c\ \theta\ u(b)]$, where c is a constant or a free variable (object in B), $\theta \in \{==, =, \approx\}$, and $u(b)$ is a function (i.e. $ident(b)$, $cont(b)$, or a user-defined function).

□

We are now ready to give the semantics of a binding system. A binding system is an automaton with events as input and action as output. It starts with B empty and forms a logically linear sequence of *create*, *delete* and *update* bindings (which are themselves atomic operations) up to the instance when the set B is queried. Its transitions maybe constrained in addition by constraints, i.e., Boolean expressions specifying consistent conditions on the state of the bindings *w.r.t.* the parameters of the operation call.

The four basic operations (*create*, *delete*, *update*, *query*) operations are supported by the proposed binding service. The design specifications are given in subsection 5.1.4. In the following, we extend the basic binding model to address consistency issues and discuss how to deal with binding failures.

4.3 A Consistent Binding Model

Definition 4.1 defines a basic binding data model. It is important to note that one cannot build a binding model that contains an arbitrary set of bindings. This leads to the definitions of a consistent binding data model.

Definition 4.7. *Consistent binding data model.*

A binding data model B_{struc} is *consistent* iff

(i) B_{struc} is a finite set,

(ii) The *ident* function is injective on B_{struc} (i.e. there is no pair of b, b', such that $ident(b) = ident(b')$),

(iii) There is a function ref_d: $B_{struc} \rightarrow D$, which associates to a binding b the objects referred to by the identifier appearing in its attributes d, so that for all $b \in B_{struc}$, $ref_d(b) = o_d$, $o_d \in D$ (i.e. for each binding, its referenced d corresponds to an object of D).

(iv) Similarly, there is a function ref_m: $B_{struc} \rightarrow M$, which associates to a binding b the objects referred to by the identifiers appearing in its attributes m, so that for all $b \in B_{struc}$, $ref_m(b) = o_m$, $o_m \in M$ (i.e. every referenced m corresponds to an object M).

□

A test on $ref_d(b)$ (or $ref_m(b)$) producing the object corresponding to d (or m) implies that the constraint holds on b. A test result of *nil* implies there is no corresponding object or *unknown* relationship (depending on application interpretations). However, if the test fails, it may or may not be the case that constraint is unsatisfied. The test may not be completed due to various failures, e.g., 1) the data or metadata site fails, 2) the data or metadata site refuses the access, and 3) the network fails. Consequently, the test may fail to verify that the constraint is satisfied. In practice, operations implementing $ref_d(b)$ and $ref_m(b)$ should report failures (including types of errors) for further investigation.

Note, $ref_d(b)$ (or $ref_m(b)$) is different from the expression $expr_d$ (or $expr_m$) (see Definition 4.1). Expressions $expr_d$ (or $expr_m$) specifies structures and representations of d (or m), which is interpreted according to implementation contexts. $expr_d$ (or $expr_m$) can not verify the existence of an object expressed by $expr_d$ (or $expr_m$).

We now define additional two binding operations to support the reference functions.

Definition 4.8. *Binding get-object operations.*

Let $\mathcal{F}$ be a set of failures. Two *get-object* operations provided by B_{behav} are given as follows:

$$\text{i) } b.getD() = \begin{cases} ref_d(b)=o_d, & o_d \in D \\ nil & \\ error, & error \in \mathcal{F} \end{cases}$$

$$\text{ii) } b.getM() = \begin{cases} ref_m(b)=o_m, & o_m \in M \\ nil & \\ error, & error \in \mathcal{F} \end{cases}$$

□

The *get-object* operations provide the means to communicate with the data and metadata sources from within the binding system.

The consistent binding data model specifies inherent consistency constraints which should be enforced by a binding system internally, i.e., operations on the binding system should preserve the consistency of the binding system that is, if the binding system is consistent when an operation starts, it should be consistent when the operation completes. Erroneous states caused by system operations should be checked incrementally.

A binding system which runs in distributed environments has complex dynamic behaviours. Maintaining binding consistency is particularly hard for an independent binding management, where the data and metadata resources are a set of pre-existing autonomous storage systems, each of which manages its own data and supports executing autonomous applications. They may be under different ownership and beyond the control of the binding system. External unknown operations can cause various binding failures and leave the binding system in inconsistent states and containing contradictory information. Recall in Chapter 2, some forms of inconsistency have been identified.

In addition to specifying internal binding data structure and restricting system operation behaviours, enforcing binding consistency is critical to satisfy various external (user-specified) consistency requirements. It should allow users to describe a set of constraints, and binding failures against user-specified consistency constraints should be detected where they are detectable.

In the following, we provide a means to formalise the practical problems and illuminate the potential usage of a binding system for managing binding consistency.

Firstly, we need to establish a sense of correctness of binding states. The binding consistency is defined via a *Correct* function.

Definition 4.9. *Correctness.* Given an instance of binding data model B, a data set D, and metadata set M, we introduce the *Correct* function: $B \rightarrow \{true, false, undecided\}$, which is in principle decidable by experts inspecting b, d, m, and a, where $b \in B$, $d \in D$, $m \in M$, and $a \in A$. It yields *true* iff the binding is *correct* at the time of inspection; *false* iff the binding is *incorrect*; *undecided*, otherwise.

□

A computable approximation to the notion of *correctness* is needed. We use (a subset of) first order logic and develop a binding constraint language. The idea is to allow users to specify consistency requirements: under what conditions the binding states are correct, and under what conditions they are not. Every expression should be explicitly defined, so that the specification is systematically analysable.

In the following, after a preliminary definition, the syntax and semantics of the binding constraint language is presented together with some practical examples.

Definition 4.10. *Binding Constraint.* A binding constraint is a statement which asserts consistency requirements for binding data and metadata.

□

The syntax of the binding constraint language is specified with BNF notations and shown in Table 4.1.

Table 4.1: Syntax of the Binding Constraint Language

I	::=	(U) → (U')
	|	U
U	::=	u ∧ w
	|	u ∨ w
	|	u *op* w
	|	¬ u
op	::=	>, <, =, ≠, ≤, ≥, ==, ≈
E	::=	$\exists x$ E, $x \in B$
	|	I
Q	::=	$\forall x$ E, $x \in B$
	|	E

In essence, the language specifies the following form of statements:

$$\forall x \exists y (A \rightarrow B)$$

The language includes the bounded quantifiers, ∀ (universal) and ∃ (existential) qualifiers, which allow one to express the query over finite domains. The restricted logical

expressions ($\wedge$, $\vee$, $\neg$, and $\rightarrow$) are sufficient and capable of expressing general binding consistency requirements. The language does not include other expressions, such as algebraic expression ($+, -, \times, \div$), and set expression ($\subset, \subseteq, \in$). However, there is no restriction to limit the language not to include these expressions. They can be added when required.

The Boolean operators, '$\wedge$', '$\vee$', '$\neg$' have the conventional meaning. A conjunction of expressions is satisfied if both sub-expressions are satisfied, a disjunction is satisfied when at least one is satisfied, and a negation is satisfied when the sub-expression is not satisfied. The implies operator '$\rightarrow$' denotes a conditional (if/then) statement. The hypothesis lies to the left of the arrow and the conclusion to the right. The operator, '==', denotes identity equality; '$>$', '$<$', '$=$', '$\neq$', '$\leq$', '$\geq$' are used for value comparisons and defined as usual; and '$\approx$' denotes value similarity.

Formally, the semantics of the binding constraint language is given by an evaluation function:

$$\mathcal{F}\colon \mathrm{Q} \rightarrow \{true, false, undecided\}$$

With the binding constraint language, the binding inconsistency problems identified in section 2.3 can now be formally addressed. The following examples look into how binding consistency requirements can be explicitly described by the constraint language.

Example 4.5. Consider the binding inconsistency *Case 1* given in subsection 2.3.2.

Suppose the binding Set B contains bindings of the associations of gene expression images in the Set D_g and the image metadata in the Set M_e. Let each binding $b_i = < id_i, d_j, m_k, \{a_{i0}, a_{i1}, \ldots, a_{ip}\}>$, where d_j refers to the image files, and m_k refers to the image metadata. Using the binding constraint language, the consistency requirement for bindings b_i can be define as two constraint statements, $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$:

$\boldsymbol{C}_1$: $\forall b, b \in B, (ref_d(b) \neq nil) \rightarrow (ref_m(b) \neq nil)$

$\boldsymbol{C}_2$: $\forall b, b \in B, (ref_m(b) \neq nil) \rightarrow (ref_d(b) \neq nil)$

The function $ref_d(b)$ gets the gene expression images in the image directory, and $ref_m(b)$ retrieves the image metadata from the metadata table.

Thus $\boldsymbol{C}_1$ states: for every binding in the binding store, if the associated gene expression image exists, the image metadata record should exists.

Similarly, $\boldsymbol{C}_2$ states: for every binding in the binding store, if the associated image metadata record exists, the image should exist.

A verification program checking with $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ can detect images without metadata record, or the metadata record without images.

□

Example 4.6. Consider the binding inconsistency *Case 2* in subsection 2.3.2.

Suppose the binding Set B contains bindings b_i, the associations of the gene expression images in Set D_g with the template records in Set M_t, and bindings b_i', the associations of the gene expression images in D_g with the ISH records in Set M_e. Thus, $b_i = <id_i, d_j, m_t, \{a_{x0}, a_{x1}, \ldots, a_{xp_i}\}>$, and $b_i' = <id_i', d_j, m_e, \{a_{y0}', a_{y1}', \ldots, a_{y(p_i+r)}'\}>$. A constraint statement can be given as C_3:

C_3: $\forall b, b \in B, \exists b', b' \in B, (ref_d(b) == ref_d(b')) \rightarrow (ref_m(b) \approx ref_m(b'))$

C_3 states: if two bindings associate with the same gene expression image, their metadata should contain information about similar genes. C_3 constrains every gene symbols used in ISH experiments must have a related template record designed for that gene. (As an example, we assume the comparisons of gene symbols in ISH records and in template records are based on content similarity, thus use $\approx$.)

□

Example 4.7. Consider the binding inconsistency *Case 3* in subsection 2.3.2.

Suppose the binding Set B contains bindings b_i, the associations of the gene expression images in Set D_g with the annotation files in Set M_a, and b_i', the associations of images in D_g with the database copy of the annotation files in Set M_a. Let $b_i = <id_i, d_j, m_a, \{a_{x0}, a_{x1}, \ldots, a_{xp_i}\}>$, and $b_i' = <id_i', d_j, m_a', \{a_{y0}', a_{y1}', \ldots, a_{y(p_i+r)}'\}>$. Similarly, consistency requirements in terms of constraints can be specified as follows:

C_4: $\forall b, b \in B, \exists b', b' \in B, (ref_d(b) == ref_d(b')) \rightarrow (ref_m(b)) = ref_m(b'))$

C_4 states: if two bindings associate with the same gene expression image, their metadata should have the same image annotations. C_4 guarantees the co-existence of each annotation file and its database copy.

□

Now we are able to formalise the notion of binding consistency checking problem.

Definition 4.11. A *Binding Consistency Checking Problem* is a 3-tuple:

$$P = (B, C, V)$$

where

- B is a instance of binding data model B_{struc};
- C is a set of constraints;
- V is an algorithm for evaluating C.

□

In the binding model, an additional operation can be defined to resolve the binding consistency checking problem.

Definition 4.12. *Binding Validation Operation.*

For B, an instance of B_{struc}, we denote $B.rules$ as binding rule set on B, which contains binding constraint statements. A binding validation operation is given by $B.validate(rule)$, which evaluates the rule against B, where $rule \in B.rules$. $B.validate(rule)$ returns *true* iff rule is not violated, *false* iff rule is violated, *undecided* otherwise.

□

Related Work

We briefly review work related to the consistency checking. The most immediately related work is research on federated databases, which addresses the problem of managing integrity constraints between heterogeneous distributed databases. [Gupt 94, Rusi 91, Gupt 93, Nent 01, Olen 90] propose languages to specify integrity constraints in a multidatabase environment. [Brei 92, Baba 93] give formalisation of the global consistency checking problems. [Ceri 93, Gref 97, Rast 93] provide approaches to consistency maintenance, for example, by using production rules, persistent queues, or constraint checking protocols. However, these approaches assume relational databases and mainly discuss how to enforce correctness of real-time transactions.

Consistency requirements are also studied in the area of distributed database replication. [Char 02] defines a consistent state and makes use of self-stabilisation theory to prove properties of eventually consistent systems. [Bosn 02] provides a formalism for specifying constraints and representing scheduling based on actions. But these approaches focus on the eventual consistency methods for database replications and do not satisfy requirements for bindings consistency.

The work on consistency management in object-oriented data models has relevance to this work. [Bene 98, Stra 03] use a descriptive formulation for type constraints and integrity rules. [Malg 06] uses Constraint Logic Programming (CLP) to develop a checker to handle inconsistencies over UML features. However these approaches focus on how to express UML classes and properties, which are inapplicable to the binding model.

The problem of verifying constraints on websites is discussed in [Fern 99] and applied in [Fern 00]. The work proposes a language that can capture many practical constraints and an accompanying verification algorithm. It is specific to the XML data model, and cannot be used to specify binding constraints.

4.4 Discussion

The purpose of developing a formal binding model is to clarify the concept of bindings, and to explore their computational properties. It also serves as a framework allowing binding problems identified in real-world situations to be described precisely. The formal model is not semantically complete, rather, it intends to be an initial effort to establish a foundation that can be extended by future research to address more specific cases for bindings.

Conventionally, relationships between entities are modelled as attributes [Grub 93, Tamm 02, W3C 04, Kim 02, Evan 98] or functions [Fan 09b, Fan 09a, Boha 06]. The proposed binding-oriented approach regards bindings as first-class citizens. It extends the Object-Oriented Data model to define bindings as independent entities, and models both static and behavioural aspects of the bindings.

The formal model describes the semantics of a binding system, and defines a binding data structure and applicable operations, in order to guide design and implementations. A design introduced in Chapter 5 is derived from these formal descriptions.

The formal model uses abstract expressions so as to include general requirements for a binding system, however, it does not restrict design choices – based on the formal model there can be different designs of a binding system. For example, by definition, a binding is a four-tuple, $< id, d, m, \{a_0, a_1, \ldots, a_p\} >$, where the expressions of d and m are denoted as $expr_d$ and $expr_m$, respectively. In a practical design, $expr_d$ and $expr_m$ can be chosen as any appropriated representations according to concrete system requirements. In the case of Example 4.1, $expr_d$ is represented by the full path of a medical image file, and $expr_m$ is represented by an SQL statement. Chapter 5 describes a design for a generic binding service, where $expr_d$ and $expr_m$ are chosen to be URIs. In the case of the OntoGrid Semantic Binding Service, $expr_d$ and $expr_m$ are in the forms of `Grid Entity` and `Knowledge Entity`, respectively. In this sense, the Semantic Binding Service of the OntoGrid can be regarded as an implementation instance of the proposed binding model.

A further design issue is how a collection of bindings is represented and whether there is one collection or many collections in a binding service. In the example implementation in Chapter 5, the representation is as a relational table and only one collection is managed per binding service. Other binding services could use an RDF store or handle many collections with an extra collection (object, topic, etc.) parameter per service call.

A binding system that implements the binding-oriented approach would be used by tools, or integrated with other services such as data storage service, digital repositories, or metadata services. Consider a service, CompositeDataService (CDS), which uses the binding services, together with a file store and a metadata store. Suppose various consistency rules are included, and the contents of the data and metadata referenced have been

changed.

On delivery of a data item o_d, a metadata item o_m, and user-supplied attributes $A = \{a_0, a_1, ...\}$, CDS would make the following actions:

1) Scan o_d and generate fingerprint f_d;
2) Scan o_m and generate fingerprint f_m;
3) Store o_d in the file store, and get reference to o_d as d;
4) Store o_m in the metadata store, and get reference to o_m as m;
5) Store a binding b with the binding service with d, m, $A \cup \{f_d, f_m\}$.

When CDS is asked to use the binding, it would act as follows:

1) Look up the binding b;
2) Retrieve o_d and generate fingerprint f_d'; Check $f_d == f_d'$;
3) Retrieve o_m and generate fingerprint f_m'; Check $f_m == f_m'$;
4) If either are not equal, then it warns the calling service or the tool of a potential inconsistency;
5) If they are equal, then return the part of the binding requested.

Similarly, the CDS could store a set of consistency rules defined in the binding consistency language. Then, it could:

1) Verify the rules periodically and add failure warnings to binding annotations which are presented when a tool or a service uses it; or
2) Run these checks when a binding is requested.

An implementation example of binding validation operation is provided in Chapter 7[2].

4.5 Summary

This chapter has presented a formal binding model which defines the key concepts and formalises binding consistency problems raised in Chapter 2. Based on this computational framework, a simple binding system is designed and developed. Chapter 5 presents the binding system and discusses issues related to system design.

[2]Discussions about the consistency checking mechanisms and optimisation algorithms are beyond the scope of this investigation. They will be postponed for follow-on work.

Chapter 5

A Simple Binding Service

Chapter 4 has defined a formal binding model and described the semantics of a binding system. However, a number of challenges for system design remain. This chapter focuses on design issues, and discusses appropriate computational representations for binding, the construction of a tagging mechanism for binding annotations, and the specification of operational APIs. To evaluate the viability of the design, a prototype service is implemented.

The chapter is arranged as follows: the specification of the binding service are presented in section 5.1. Section 5.2 describes the prototype implementation. Section 5.3 discusses the effectiveness of the binding approach, and section 5.4 summaries the chapter.

5.1 Design

We are designing a prototype experimental binding service to test the thesis hypothesis. The design concentrates on specific aspects (including the binding representation and operation APIs), and separates these aspects from other concerns (such as some engineering details) in the development of a production binding system, thus reducing design complexity.

5.1.1 Design Assumptions

The design of the binding system is based on a number of assumptions:

- The binding services are to be used by various types of user-defined applications in a variety of computational environments.
- For different users, data and metadata can mean different things: some users regard the annotation terms (e.g. creator, titles, and descriptions) as the metadata, and the database entries as the data; some users treat the database entries as the metadata

(e.g. the biological experimental data providing the background information of the output images, for biologists, they are the metadata of the images), and the digital media as the data. The computational representation of the bindings should be neutral to user's view of data and metadata. Nonetheless, the design of the binding system uses binding *subject* and *object* to indicate the data and the metadata, respectively. Distinct only in order, the representations of them are essentially the same.

- When in use, bindings are assumed to be "Write Less Read Mostly (WLRM)". Therefore, in certain situations, in order to optimise binding-read performance, binding-write performance may be sacrificed.

- Security will be dealt with by calls to other services. In reality, security efforts are often application and organisation dependent, and users tend to employ their own security policies. Security can adequately be provided by the application using the binding service. Once requested, security features can be added-on to the binding service, which will not change the design principles significantly.

5.1.2 Design Overview

To design a binding system, we need to carefully examine each potential system property, only add necessary functionality, and remove controversial or conflicting elements. This can reduce the internal computing complexity.

The binding system should contain no redundant component that might overlap with any subcomponents of a coupled service, or could be provided by a separate application, such as user authentication, replication, or monitoring facilities. This will help it to be integrated with a range of applications with minimal difficulties in systems integration.

Consistent with the formal model, the design of binding system focuses on two key components: a binding data structure, and the operation APIs.

The binding data structure provides the computational representation of the bindings, which should be independent of the data or metadata objects, the resources, the application end point or the underlying communications protocols and service model as far as possible. It should be able to support heterogeneous data and metadata representations and capture the dynamic changes among them.

The binding operation APIs provide the behavioural functionality, which should be as simple as possible to ensure scalability [Cudr 08]. A minimal set of binding operations are specified based on the formal definitions. More sophisticated binding operations may be delivered by tools and services using these primitive binding functions. However, it may necessary for performance or other reasons to offer some of these advanced operations

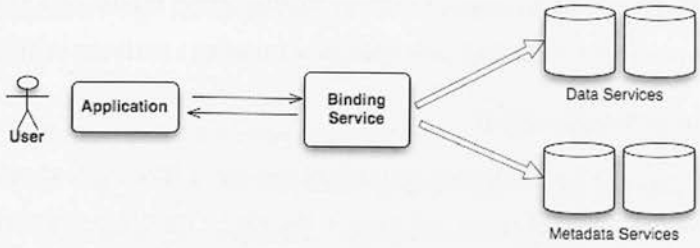


Figure 5.1: Binding Service Communication Framework

from within the service. Such extensions should be investigated once they have been proved necessary.

The binding system is chosen to be web service style. The communications between the binding service and its computing contexts are illustrated in Figure 5.1. On the one end, a deployed binding service persistently interacts with the application end-point and responds to requests by delivering computation results or exchanging messages; on the other end, it is linked (virtually) with the data provider, the data services, and the metadata provider, the metadata services, via references.

5.1.3 Data Structure

Recall in Chapter 4, Definition 4.1 has defined the notion of binding. Accordingly, a computational representation for binding is derived and graphically depicted in Figure 5.2. The proposed data structure for the binding service consists of a global unique identifier, BindingID, a reference pointing to the data (the so-called binding *subject*), a reference of the associated metadata (the so-called binding *object*), and a set of tags that are used to describe other attributes or annotate the semantics of the binding.

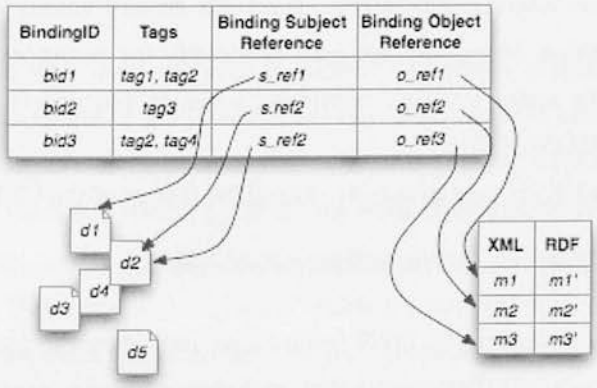


Figure 5.2: Binding Service Data Structure

By definition, binding data and metadata can be *nil*, thus NULL is introduced in the representations of the references. The practical reason for this is that the real-world datasets

usually contain a certain percentage of missing values, and a robust data structure should comprise some routines for dealing with unknown (missing) attribute values [Bruh 04].

Representation of the BindingID

For a binding service in a global-scale distributed system, a BindingID should have a high likelihood of uniqueness over space and time, it should be locally generated without contacting a global registration authority, and be able to reliably identify a potential huge number of binding objects across a network.

For above reasons, ISO standard [ISOI 01, Leac 05], the Universally Unique Identifier (UUID) is chosen as the systematic representation of the BindingIDs. The format of a UUID consists of 32 hexadecimal digits, displayed in 5 groups separated by hyphens, for example:

73b2a8ee-dd76-47eb-962c-08f5430f313d

128 bit in total, therefore, the number of theoretically possible UUIDs is 2^{128}, which means that 1 trillion UUIDs would have to be created every nanosecond for slightly more than 10 billion years to exhaust the number of UUIDs. Depending on the specific mechanisms used, a UUID is almost guaranteed to be different from any other UUIDs. UUIDs can be created by generation schemes such as MD5 hashing, random number generation, or SHA-1 hashing, based on local clock sequence or IP address.

Representation of the References

The Uniform Resource Identifiers (**URI**s) structure is adopted as the representation of the references of the data and the metadata in the binding data structure. Most resources currently on the World Wide Web: documents, images, downloadable files, services, electronic mailboxes, are referenced by URIs. The URIs defined within W3C globally distributed communication framework are used to identify the resources on the web, and make them available under a variety of naming schemes and access protocols such as HTTP, FTP, and SSH [W3C 01].

The syntax of a URI is easy to specify. Recall the format of the URI [Bern 05].

<scheme name>:<hierarchical part>[?<query>][#<fragment>]

A URI consists of four parts: (1) Scheme name, indicating the connection protocol, e.g., ftp, http, urn, ssh. (2) Hierarchical Part, including authority (user information, host name, port number), and path (e.g., the file path), (3) Query, contains non-hierarchical data, which enables the representations of data objects in different level of granularities, e.g., using a query can select the whole datasets of a database table or locate a single data item, and (4) Fragment, allows indirect identification of a secondary resource by reference

to a primary resource, e.g. an anchor place for where the page should be displayed. As examples:

ftp://gene.ish.lab/head/gene/image.png

– ftp scheme for File Transfer Protocol services

http://www.gene-expression.org/db/query?anatomy=olfactory

– http scheme for Hypertext Transfer Protocol services

URI syntax includes the representation of empty, which contains only the scheme component. This is used as the representation of a NULL reference in the binding data model.

A number of popular identifier systems defined as URI schemes that use URI syntax include URL (Uniform Resource Locator), URN (Uniform Resource Name), URC (Uniform Resource Characteristics), DOI (Digital Object Identifier), and LSID (Life Science Identifiers). A **URL** is intended to be used for locating or finding resources, and generally used to describe a resources current location [Meal 02]. A **URN** is intended to serve as persistent, location-independent, resource identifiers. It refers to the subset of URI that is required to remain globally unique and persistent even when the resource ceases to exist or becomes unavailable [Soll 99]. **URC**s were proposed as semi-standard form of metadata that described web resources identified by a URI[1] [Soll 99, Meal 99]. A **DOI** is a permanent digital identifier adopted by many publishers [ISO 08]. The DOI organisation is registered as a part of URI within the info-URI namespace as a *doi* scheme. The **LSID**s are persistent location-independent, resource identifiers for uniquely naming biologically significant resources, LSID is built on top of URN [Life 05]. Therefore, by using URIs, a binding system is capable of interacting with a majority of representations of the web resources.

While URI can well serve the need to federate identifiers of almost all types of resources in different levels of granularities on the web, there are some drawbacks of it (e.g., that a URI cannot contain another URI, nor XML), which have led to the introduction of a different identifier system by W3C, the **EPR** (Endpoint References) [Gudg 05]. This type of resource is not handled by the binding system. However, EPRs are mainly used to reference Web Services endpoints, which are the user devices connected to the network, such as firewalls, gateways and endpoint managers, therefore, they are not essential for this implementation. Addressing the EPR representation could be included in future work.

[1]The effort to standardise URCs within the IETF gave up after its initial introduction. The need for standard metadata description of a resource has been solved by various technologies such as Resource Description Framework (RDF) and Meta tags.

Representation of the Annotation

A pervious chapter (in section 3.3) has introduced the tagging technology which is one of the simplest annotation frameworks, human-friendly and platform-independent. The binding service uses tags for the representation of binding annotation.

Depending on underlying systematic processing mechanism, a tag can be a single-word, a phrase, a key/value pair, a list, a block, or a reference. In general, the single-word tag is efficient for cataloging and classification, while the tag containing phrase is useful for description. For example, since Flickr does not support the phrase inputs, in its Hot Tags List, there are many tags like 'thunderoverlouisville', 'day112', 'fridaydesignface', which shows the difficulty for the users to find one meaningful single word for description. On the other hand, the key/value-pair tag, for example, 'height=17.3', 'length=14.2', 'width=56', is often required when the calculation or comparison is crucial. Besides these string-based tags, there are non-string-based tags existing, i.e., those contain XML fragments, blobs (binary large objects), or nests of another list of tags. The non-string-based tags are normally generated automatically. Such machine-generated tags are potentially large in quantity and complex in data type and structure.

Tag inputs are often in series and form a collection of tags. The tag collection can be accommodated within a *set*, a *bag*, or a *sequence*. A *set* is a collection of distinguished elements where the order of elements is irrelevant; a *bag* is a generalisation of a *set* where item duplications are legal; and a *sequence* is an ordered *bag*, in which the elements can be repeated but are listed in order. Given the same tag input, the tagging systems implementing *set*, *bag*, or *sequence*, may store and output tags in different orders and numbers. For examples, for a tag input:

{'red', 'red', 'blue'}

A tagging system implementing tag *set* will remove the duplicated items, and return an output as:

{'red', 'blue'} or {'blue', 'red'}

A tagging system implementing tag *bag* may discard the original order of the input, and output:

{'red', 'red', 'blue'} or {'blue', 'red', 'red'} or {'red', 'blue', 'red'}

A tagging system implementing tag *sequence* will rank the tags in (alphabetical-) order, and output a list as:

{'blue', 'red', 'red'}

As the result, the tag *sequence* does not suit applications where the orders of the input tags are meaningful.

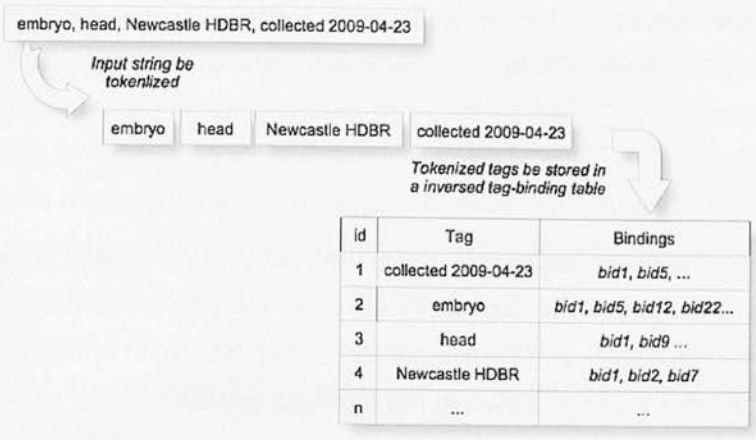


Figure 5.3: Binding Tagging Mechanism

In the design of binding service data structure, the mathematical meaning of *set* is chosen to process tags, since we want to discard the order of the input tags (thus, a tag *sequence* is unsuitable), and we also want to remove the duplications from the input tags (thus, a tag *bag* is unsuitable). The binding tagging mechanism is shown in Figure 5.3. Initially, the tagging API is implemented as String Array type (String[])[2]. Discussions of other tag structures, such as key/value pairs, blobs, or lists, are omitted. These can either be implemented as an extension in future implementations, or as a plug-in function from a third-party's software, such as YAML[Resn 01], which provides facilities of parsing and interpreting tags.

Some scientific systems require the semantics of tags to be explicitly defined. Depending on concrete requirements, the semantics of tags can be specified by combining with a domain ontology, controlled vocabulary or specific metadata standards.

The process of separating tags from the tag list, so-called tokenization, is intentionally left for the application layer. Tokenization is application-dependent. For example, some applications may implement a single text box accepting one tag each time; some may use an input fields allowing continued input of tags with specified separators, such as COMMA or SPACE; some others may use XML or RDF to contain tags. Leaving the tokenization with the application side will maximise the flexibility for a tagging editor.

The tags are stored in an inverted tag-binding table. Indexing by tags can ease the tag-based queries and statistical analysis. For example, it is efficient to answer a query like, "give me the bindings annotated by any of these terms, embryo, head, nose." or "tell me the most used tags".

In addition to user or application supplied tags, a group of system tags is defined. The binding system tags are a set of configurable tags attached to a binding automatically by

[2]Apply (Java) Set operations to prohibit the duplicated elements.

the system. Four system tags are specified, mainly used for administration purposes, and they are: CreationDateTimeStamp, CreationUser, LastModificationDateTimeStamp, and LastModificationUser.

Discussion

An alternative binding data structure can be considered. As illustrated in the left diagram in Figure 5.4, a binding can be viewed as a metadata-data pair, where the representation of metadata can be either tags or reference, and the representation of data can be either reference or another binding. Although this approach appears to make a further step of simplification, in practice, it results in the nested structure, as shown in the middle diagram in Figure 5.4, which is inefficient for the search and scan. As one of the assumptions clarified in section 5.1, bindings are to be read more and written less, this approach has been deliberately abandoned to ensure a graceful binding search performance.

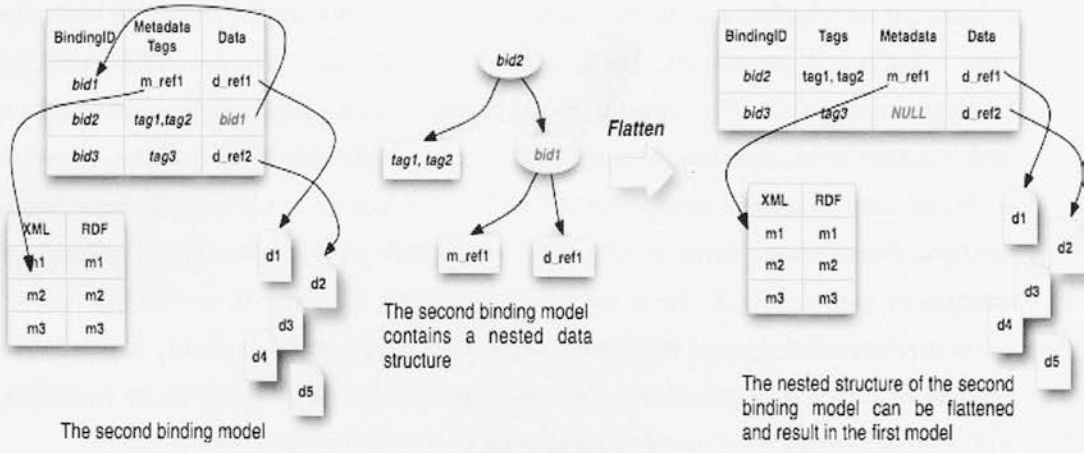


Figure 5.4: An Alternative Binding Service Data Structure

Indeed, when the nested structure is flattened, two structures become identical. Illustrated in the right diagram in Figure 5.4, tag is separated from metadata and become additional attribute which contain no references. The binding bid1 is merged with bid2, that is, binding bid2 takes bid1's metadata as metadata and bid1's data as data. The binding bid3 is extended to have the metadata attribute (the value of which is Null).

5.1.4 Operations

Chapter 4 has described the semantics of binding behavioural operations. This subsection discusses how to construct binding service APIs to support those operations. To improve the modularity and support the reuse of codes, the binding APIs are organised into four groups: manipulation, discovery, communication, and validation.

The manipulation APIs related to binding creation and updating behaviours include: CreateBinding(), DeleteBinding(), AddBindingSubject(), AddBindingObject(), InsertTags(), RemoveTags(), and UpdateTags(). Among them, the CreateBinding() API is derived from Definition 4.3, the binding *create* operation; the DeleteBinding() API corresponds to Definition 4.4, the binding *delete* operation; the functions of binding *update* operation described by Definition 4.5 are separated into five APIs: AddBindingSubject(), AddBindingObject(), InsertTags(), RemoveTags(), and UpdateTags().

Four discovery APIs are specified to support the binding *query* operation (Definition 4.6), including: GetBindingByID(), GetBindingsBySubject(), GetBindingsByObject(), and GetBindingsByTags().

The communication APIs correspond to Definition 4.8, the binding *get-object* operations, used to interact with the resources and obtain the data entities or the resource status from remote sites, including: GetResource() and GetResourceStatus(). Note, the two binding *get-object* operations are specified through one API, GetResource(), because the implementation algorithms for the two operations are the same.

One validation API, VerifyBindings(), is specified corresponding to Definition 4.12, the binding *Validation* operation, which is used to check bindings against user-defined constraints.

Table 5.1: The Effects of Binding Operation APIs to the Binding States

Operation API	**Description**	**Change of State of Binding**
CreateBinding ()	Create a new binding	New Binding
DeleteBinding ()	Delete a binding	Delete
AddBindingSubject ()	Add the binding subject reference if previously is null	Update
AddBindingObject ()	Add the binding object reference if previously is null	Update
InsertTags ()	Insert binding tags	Update
RemoveTags ()	Remove binding tags	Update
UpdateTags()	Update binding tags	Update
GetBindingByID ()	Return a binding by matched BindingID	Unchanged
GetBindingsBySubject ()	Return bindings by matched subject	Unchanged
GetBindingsByObject ()	Return bindings by matched object	Unchanged
GetBindingsByTags ()	Return bindings by matched tags	Unchanged
GetResource ()	Obtain binding subject/object	Unchanged
GetResourceStatus ()	Obtain status of binding subject/object	Unchanged
VerifyBindings()	Check bindings against consistency constraints	Unchanged

The effects of the binding operations to the binding state are listed in Table 5.1 which shows that the invocations of the manipulation operations have side-effects on the binding state, while all discovery and communication operations are side-effect free. In the rest of this subsection, the specification details of each operation are described in turn.

Manipulation APIs

From subsection 5.1.3, the definition of binding type, B, can be derived:

{
binding_id: UUID
ref_sub: URI
ref_obj: URI
tags: String[]
}

The APIs of the manipulation operations are specified below. All operations are completed in one of three status: 1) no effect (when execution successes, but 0 records are updated); 2) the number of records updated (when execution successes, n records are updated); or 3) errors (when execution fails).

CreateBinding(*ref_sub*: URI, *ref_obj*: URI, *tags*: String[]);
DeleteBinding(*binding_id*: UUID);
AddBindingObject(*binding_id*: UUID, *ref_obj*: URI) ;
InsertTags(*binding_id*: UUID, *tags*: String[]);
RemoveTags(*binding_id*: UUID, *tags*: String[]);
UpdateTags(*binding_id*: UUID, *old_tags*: String[], *new_tags*: String[]);

The CreateBinding() API accepts the inputs of a reference to the binding subject, a reference to the object, and a list of tags. It generates a new UUID type BindingID, and creates a new binding entry, then invokes InsertTags() to update the binding tags.

The DeleteBinding() API removes a binding entry having the indicated BindingID. It also invokes RemoveTags() to remove all mappings of the BindingID and tags.

The AddBindingSubject() and AddBindingObject() APIs are applied to the bindings containing Null references. This pair of operations complete the binding-creation behaviour initiated by a CreateBinding() operation which has generated a binding entry with a missing subject or object value. (The executions return errors if the binding subject (or object) is non Null.)

The InsertTags() API checks the tag index, and inserts the tags if they are new. Then, it creates the mappings of the indicated BindingID with each input tag. A reverse behaviour, RemoveTags(), removes the associations of the given BindingID and tags. But it does not remove tag entries from the tag index, since those tags may be reused by other bindings. Finally, UpdateTags() replaces the indicated tags with new tags.

Note, only the means for updating binding tags are provided, and the modifications for any (non null) binding objects or the binding subjects are prohibited. The philosophy is that the change of the association of two objects are regarded as a new relationship, thus a new binding object should be created in the system. This may also benefit binding provenance, since every change of an individual binding subject or object can be preserved, thus traceable.

Discovery APIs

The four discovery APIs are given as follows:

GetBindingByID(*binding_id*: UUID)
GetBindingsBySubject(*ref_sub*: URI);
GetBindingsByObject(*ref_obj*: URI);
GetBindingsByTags(*tags*: String[]);

The operation, GetBindingByID() fetches the binding having matched BindingID. For a given reference value, GetBindingsBySubject() returns bindings with matched subject. Similarly, GetBindingsByObject() returns bindings with matched object. The wildcard (_ , %, *) access is supported, which can substitute for one, more or all characters of the lookup reference. For example, GetBindingsBySubject("*ftp: //gene.ish.lab/head/gene/image% .%*"), will retrieve bindings having subject associated with any files in the indicated location with name start with '*image*'.

The GetBindingsByTags() operation takes multiple tag inputs, and retrieves the bindings containing **any** of specified tags, that is, the relationship between the input tags is assumed to be Logic-OR. Similarly, pattern-matching is allowed.

These four discovery operations are intended to provide a set of atomic functions, allowing an external application to access the binding information. From these basic building blocks, more sophisticated search facilities can be assembled, for example, by performing union or intersection over the returned results, sorting or aggregating the result sets, or embedding the operations within control flows, such as sequence, choice, and iteration.

As an example, consider a query requirement: "give me the bindings having subject ref_sub_x, AND object ref_obj_y OR ref_obj_z". Which can be achieved by the following sequence flow:

1: B_1 = GetBindingsBySubject(ref_sub_x);
2: B_2 = GetBindingsByObject(ref_obj_y);
3: B_3 = GetBindingsByObject(ref_obj_z);
4: $B_{result} = B_1 \cap (B_2 \cup B_3)$;

Communication APIs

Two operations are provided for binding communication:

GetResource(*ref*: URI);
GetResourceStatus(*ref*: URI);

For a given non null reference value, GetResource() retrieves the data item from the remote data resource by using specified protocol, and GetResourceStatus() checks and returns the resource status which can be unchanged, modified or unavailable.

Validation API

Finally, one binding validation API is specified:

VerifyBindings(*c*: String);

VerifyBindings() takes a binding constraint statement specified in the binding constraint language which is defined in Chapter 4, and parses it into executable expressions. It returns *true* if the translated statement is evaluated as valid, *false* if the statement is evaluated as invalid, or *undecided* if the execution fails, in which case information is provided to allow a researcher to trace the errors. In Chapter 7, an implementation example and an experimental evaluation of the binding validation operation will be presented.

5.2 Implementation

To demonstrate the viability of the design, a web service is implemented, which is built on top of the Grid data access and integration middleware, OGSA-DAI[3]. OGSA-DAI provides useful facilities to ease the development of web service for distributed data access and movement [Anto 07]. In its newly released versions, OGSA-DAI presents neater structure, upgraded strategies and enhanced performance for access to distributed data.

Figure 5.5 depicts the architecture of the binding web service. Each binding service consists of a binding store which is made web-accessible by OGSA-DAI. Within the OGSA-DAI framework, the binding operations are implemented as the OGSA-DAI server activities and client activities which are individual units of work in a workflow that perform a user-defined task, such as running an SQL query to create bindings, look up bindings, or retrieve data or metadata from remote resources. OGSA-DAI supports remote data retrievals via HTTP, FTP or GridFTP protocols.

The prototype binding service uses a relational database to store bindings and uses SQL language for query. As shown in Figure 5.6, the relations schema consist of: BindingTable, TagTable, BindingTagTable and Binding_(**datatype**)_KeyValuePairTagTables. BindingTable stores the information of binding IDs, the references to data and the reference to the metadata. The binding tag information is stored in TagTable (which serves as a dictionary of tags), BindingTagTable (which stores the mappings of tags and bindings), and Binding_(**datatype**)_KeyValuePair TagTables. In the relational database systems, different data types are associated with different operations. To facilitate type-related computations, for each different data type, a Binding_KeyValuePairTagTable is defined to store the tag-values based on the tag-value types. Tables are indexed: BindingTable is indexed on BT_bindingID, BT_subject, and BT_object; TagTable is indexed on TT_tagID,

[3]OGSA-DAI: http://www.ogsadai.org.uk/

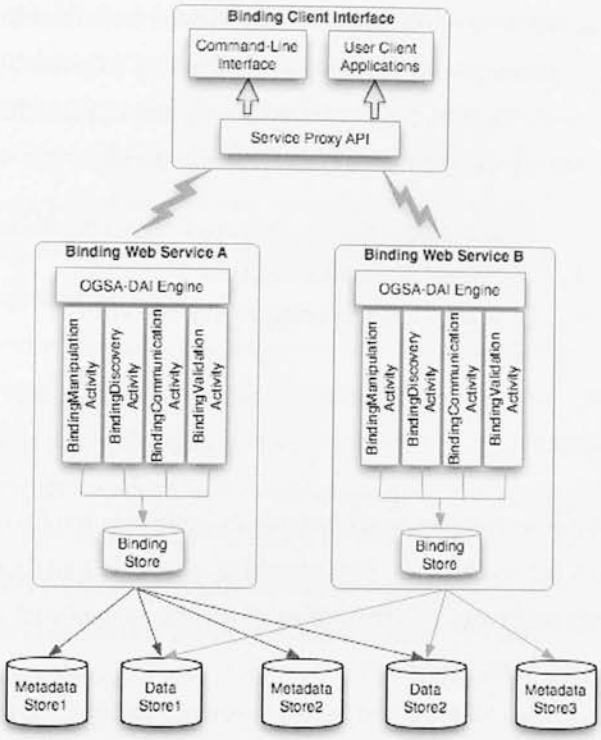


Figure 5.5: Binding Service Implementation Framework

and TT_tag; BindingTagTable is indexed on BTT_bindingIDs and BTT_tagID, and Binding_(**datatype**)_KeyValuePairTagTables are indexed on B*TT_bindingID and B*TT_tagID. (See Figure 5.6 for table attributes.)

Two client interfaces are provided: a binding service proxy API and a command-line interface. The binding service proxy API is a programmable interface, providing the access to a binding service located on a remote OGSA-DAI server. The user's client application can be connected via this interface.

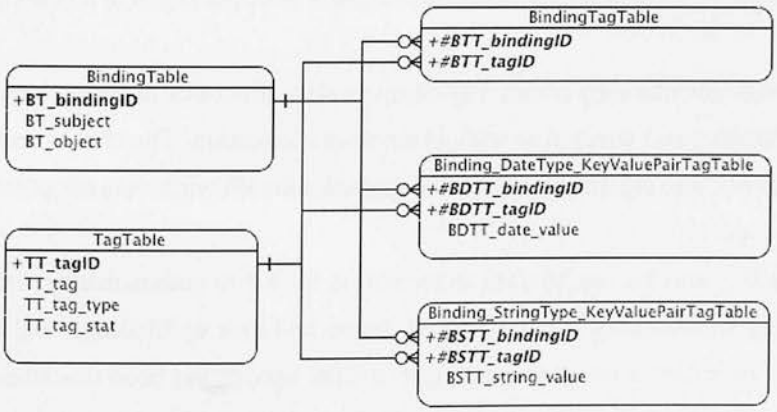


Figure 5.6: Binding Service Database Schema

A command-line interface allows manual interactions with the binding service. For example, a command-line input for creating a new binding is given below. Such an interface is intended for developers, e.g., to initialise a system. It is unlikely to be used by application users except prepackaged in a command-line script.

```
> Create   -s   "http://www.example.com/embryo/embryo102_olfactory.png" \
           -o   "http://www.genebank.com/gene/" \
           -t   "gene expression detected, strong, important"
```

5.3 Discussion

The objective of the design has been to investigate a simple solution for the management of scientific data and metadata bindings. This should be a generic tool that can be integrated with a variety of software tools, services, or applications, to deal with scientific data and metadata in various forms and volumes.

Aiming at these goals, the proposed binding service has been shown to be a practical computational representation for bindings, accessed by a set of applicable operation APIs. Firstly, using UUID allowed local generation of globally unique binding identifiers without consulting a central server. Having globally identifiable binding IDs is useful, for example, in the process of integrating two or more sets of bindings from distributed sources. Secondly, using URIs allows bindings to reference a majority of web-accessible data and information. This data reference approach avoids data and metadata being copied into central storage and enables the management of bindings to be independent. Finally, using tags allow different types of user applications to describe bindings in various contexts. i.e. a provenance application would have tags, 'ComeFrom=Server 1', 'Function=Scan', and a biological image annotation application would have tags, 'Embryo', 'Abnormal'. In a sense, the design provides a data service as easy to use as Flickr, in other words, a Flickr-for-data. (This would require the functionality to be packaged in an easy to use web portlet.)

The design intentionally avoids any complex data structures in order to reduce costs in implementation and integration with an external application. The chosen technologies, i.e., UUID, URI, and tag, have been wildly applied, thus allowing easy adoption by application developers.

The binding service can be used as an add-on toolkit to automatically extend an existing system with binding facilities, i.e. to create and look up bindings, and to validate consistency of information. As an example, a CDS service has been described in a previous chapter (see section 4.4). In a later chapter (Chapter 7), a sample installation of the binding service to a real-world scientific system will be introduced. The example will

demonstrate how the binding service can be used to detect realistic errors within EurExpress data and metadata. Such a validation facility is not automatically provided in related work, i.e., the OntoGrid Semantic Binding Service, or a Linked Data application. In Chapter 7, we will also discuss binding solutions to various other use cases. In particular, we will revisit the seven scenarios resulting from the primary study and illustrate how to use the binding service to support those requirements including, annotation, data reference, data sharing, and data publication.

The design deliberately chooses to denote data and metadata by referencing them in other storage services. This choice met an issue experienced by chemo-informaticians who were not allowed to copy data and metadata they wished to annotated. However a disadvantage of this arrangement is the need to compose the binding service with storage services and suffer the extra response delay of calls to those services. This complexity can be hidden but the effects on network traffic and response times remain. Of course, an implementation of a binding service which can also store data and metadata could be provided for 'production' use. This additional implementational complexity was avoided as, although it is important in practice, it is not pertinent to this thesis.

5.4 Summary

The design of the binding service for supporting scientific data and metadata has been presented. A computational representation of binding has been proposed, consisting of a global unique binding *identifier*, the *subject* and the *object* that reference data and metadata resources respectively and user-assigned tags that describe the nature of the binding relationship. A set of service operations has been defined. The design has been demonstrated by a prototype implementation. The next chapter evaluates the performance of the binding service.

Chapter 6

Performance Evaluation

In this chapter, controlled experiments are organised. The results will confirm (or otherwise) the hypothesis that the binding approach is feasible to serve scientific data and metadata. The experiments also aim to identify the limits of the system, any performance bottlenecks, and reveal opportunities for optimisation. These may influence design choices in future work. Characteristic workload of creations and queries that such a service must support are reported. Then, synthetic workloads are simulated and the efficiency of the binding service for handling simulated workloads are measured. To provide a solution for scaling up the binding system, potential advantages of Cloud technology is explored, and the performance and scalability is quantified by experimental measures.

The chapter is arranged as follows: Section 6.1 evaluates the feasibility of the binding service. Section 6.2 evaluates the scalability of the binding service. Section 6.3 summarises the chapter.

6.1 Experimental Evaluation of Feasibility

Performance evaluation methods can be classified into three main categories: simulation methods, analytical methods, and monitoring methods [Jain 91, Krek 04]. The simulation approaches divide further into hardware simulation, functional simulation, and workload simulation. The analytical methods are typically used in resource organisation design and capacity planning, i.e., the determination of computing, caching and buffering capacity. The monitoring approaches are typically applied to working prototypes, or final systems [Krek 04].

For quantitative evaluations of the performance of the binding service, we use a model-driven workload simulation method. Workloads capture execution behaviours of a use case. A use case is composed of executed applications and system software, which consist of functions and their control and data flow structures [Krek 04]. Workload modelling and simulation requires to create models that represent the characteristics of observed

workloads as closely as possible, and uses a computer program to generate estimated workload traces based on the model. This method allows us to examine the performance behaviours of the prototype service in supporting simple to complex use cases before a production binding service is available.

The workload simulation method involves two steps. Firstly, workload modelling is required to produce simulations that are likely to be predictive of those of a deployed system. To this end, system monitoring information of data and metadata generation and access behaviour from scientific systems are gathered from World Wide Protein Data Bank (PDB)[1], the British Atmospheric Data Centre (BADC)[2], EurExpress project[3], and NanoCMOS project[4]. Secondly, a workflow simulation environment is set up to generate representative workloads based on the real-world observations. Therefore, the correlations of workloads and service performance, in terms of productivity and efficiency, can be observed and measured.

In the following, the workload modelling, experimental setup, and experimental results are presented.

6.1.1 Experimental Workloads

As the proposed binding service is (to the best of our knowledge) the first application federating the associations of the data and metadata, there are no readily available statistics for a system of this type. However, relevant statistics on queries and updates are available from a number of scientific and Web 2.0 systems.

Finding suitable workload data is not easy. Many scientific data repositories do not provide such statistics, since to collect this information requires extra implementation and management effort. A number of systems keep the accounting or activity logs from which relevant information can be extracted, however, most of them are for private use only. There are some published statistics, however, these do not always contain the required level of detail. An extensive search of repository sites has uncovered the four suitable data sets. Sometimes, one can inquire of projects about their system logs, as the efforts we made with EurExpress and NanoCMOS. Care has to be taken, since such logs always contain users' information or other sensitive data, therefore privacy could become an issue.

Binding Write Workload Patterns

Three types of data and metadata creation workloads are identified: *Uniform Small Updates*, *Continuing Growth* and *Intensive Burst with Uniform Background*. These are the binding *write* workloads.

[1] wwPDB: wwpdb.org
[2] BADC: badc.nerc.ac.uk
[3] EurExpress project: www.eurexpress.org
[4] NanoCMOS project: www.nanocmos.ac.uk

A. Uniform Small Updates. Uniform small updates are seen in the data collected by the EurExpress project. A previous chapter has introduced the project background: briefly, the EurExpress project high-throughput generates the expression patterns of ~20,000 genes via RNA *in situ* hybridisation (ISH). These are recorded from sagittal histological sections from 14.5 days wild type mouse embryos, and result in a digital 'transcriptome atlas'. Each has detailed descriptions of gene expression patterns in terms of the underlying anatomy. A data warehouse has been built to federate the gene expression images and the related metadata including the experiment information and annotations. The annotations are made by biologist experts manually, and stored in XML. (See section 2.3 for a detailed discussion of this project's binding requirements.)

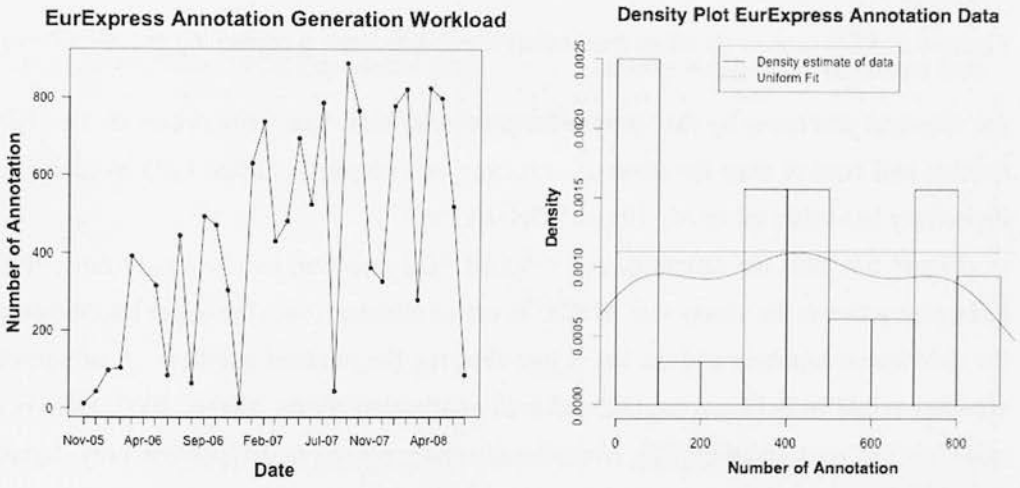


Figure 6.1: EurExpress annotation generation workloads show a uniform pattern.

The left graph of Figure 6.1 is the plot of daily annotation workload trace from November 2005 to May 2009, and the right graph is the histogram. The histogram shows that the frequencies of the number of annotation are relatively flat across the range of the data. This suggests that the uniform distribution might provide a better distributional fit than the normal distribution. We use uniform distributions to model this workload. Compared to automated processing, human behaviour tends to be stationary. To model this workload, a simple Uniform Distribution is used:

$$f(x) = 1/(b-a), a \leq x \leq b \tag{6.1}$$

B. Continuing Growth. The continuing growth pattern is observed from the statistics of protein structure deposition provided by the wwPDB site. The World Wide Protein Data Bank (PDB) maintains a single protein data bank archive of macromolecular structure data that is publicly available to the global community. The PDB structures are submitted in

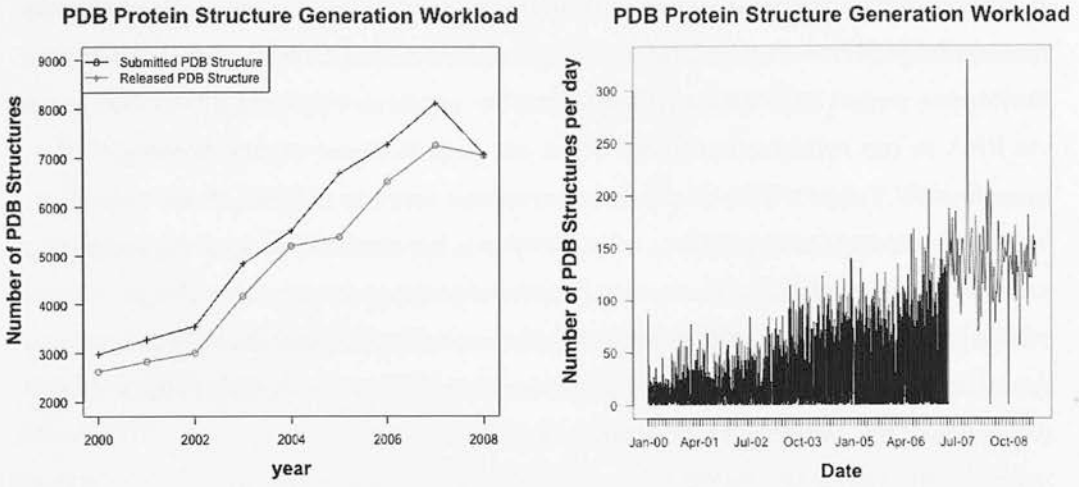


Figure 6.2: PDB protein structure generation workloads show a continuing growth pattern.

flat files and processed by the 'archive keepers' who have sole write access to the PDB archive and control over the directory structure and contents. Since 1973 to date, the depository has collected nearly 70,000 PDB structures[5].

Figure 6.2 plots the submitted and released PDB structure numbers over time. The left figure presents the yearly sum of PDB structure numbers, with the upper line showing the submission numbers and the lower line showing the released numbers. A submitted structure might be held and not released until publication. From 2000 to 2007, there is a clear trend of continuing growth. Note, the accumulated data in 2008 covers only part of the year. When plotting the daily released PDB structure number, as shown in the right figure, we still see the linear growth trend from 2000 to 2007 which means the workload changes in a well-defined way with time. This workload appears to be common for many scientific data repositories. The increasing linear trend might due to the reasons 1) the system management becomes more efficient, 2) the site gained more popularity, and 3) technologies to perform the biological experiments being upgraded.

Modelling this workload is straightforward. For a simulator to generate binding-write workloads, a certain increment is added with each time unit, t, thus have:

$$X_t = A_0 + A \times t + \xi_t \tag{6.2}$$

where A_0 is the starting workload, A represents the deterministic linear trend and ξ_t is a random fluctuation that occurs at t.

C. Intensive Burst + Uniform Background. A third data generation workload pattern is discovered from the workload traces of the NanoCMOS project and the Atmospheric Chemistry Experiments from BADC.

[5] http://www.pdb.org/pdb/static.do?p=general information/pdb statistics/index.html, retrieved Dec. 2010

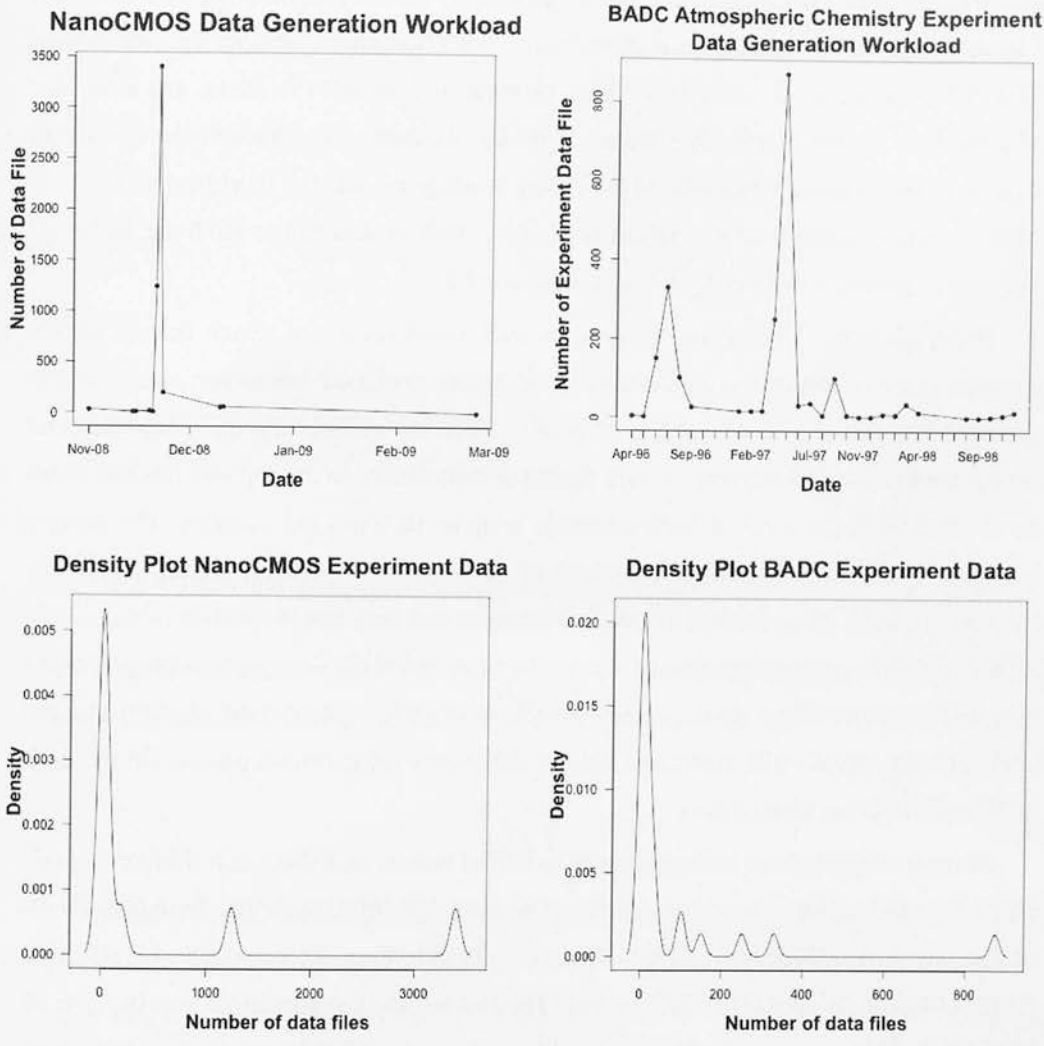


Figure 6.3: NanoCMOS and BADC experimental data generation workloads show a burst pattern.

The EPSRC pilot project, NanoCMOS started since October 2006, applies e-Science and Grid technology to investigate new architectures for integrated circuits and systems to capture the complexity of the increasing device variability in Complementary Metal Oxide Semiconductor (CMOS) transistor designs [Sinn 07]. Users submit jobs to the Grid service and conduct higher-level circuit and system simulations. The simulation results together with the metadata in file formats are generated and stored for the later analysis. The top-left figure of Figure 6.3 describes 5 months (November 2008 ~March 2009) workload trace of NanoCMOS experimental data generation.

The British Atmospheric Data Centre (BADC) is the Natural Environment Resource Council's (NERC) designated data centre for the Atmospheric Sciences. The BADC data collection includes ground based observations from Met Office surface stations, Satellite

data from TOMS, Envisat & MSG, submissions from NERC programmes such at UTLS, CWVC & URGENT, and model outputs from NWP (Numerical Weather Prediction) and ECMWF re-analyses & Climate models. Around 130 datasets in store, and total size of collection is over 60TB. One theme of BADC datasets, the Atmospheric Chemistry Studies in the Oceanic Environment (ACSOE), is analysed, and the workload statistics of ACSOE experimental data generation from April 1996 to December 1998 are gathered. The plot is shown in the top right figure of Figure 6.3.

The large-scale flurries are observed in both workload traces. Such flurries do not span the space of attribute combinations of the whole workload, but rather concentrate at a certain point or at a small number of points. From the plots of the daily experimental data generation workloads over certain period of time shown in the top two figures, it can be observed that the level of burst activities is up to 10 times the average. The bottom two figures show the density distributions of the observation values, which expose the existence of large proportion of low deposition rates and very low proportion of extremely high deposition rates. The reasons for the burst in scientific data generation processes may due to batch arrivals (in the case of BADC workload trace), or some special users (or applications) occasionally come and behave differently from common users (in the case of NanoCMOS workload trace).

Several methods have been proposed to model bursts, including self-similarity modelling [Liu 08], Markov-modulated Poisson process (MMPP) [Musc 05, Shah 00], Batch Markovian Arrival Process (BMAP) [Klem 03], Pseudo-Periodic simulation [Li 07], and Hidden Markov Models (HMM) [Feit 02]. The Hidden Markov Models, a specific type of Markov model, is used by the binding workload simulator. HMM modelling includes two parts: the first is a Markov chain, which provides the dynamic, i.e., how things change with time; the other is a set of distributions for each state in the Markov chain that generates the outputs [Feit 02]. The binding workload simulator defines N states of Markov chain, the distribution of generated bindings at each state is $V = \{v_1, v_2, \ldots v_n\}$.

Binding Read Workload Patterns

To obtain data and metadata access patterns for representing the binding *read* workload, two scientific systems have been observed. Figure 6.4 shows the access workload traces of PDB data from August 2007 to April 2009 (in the left figure), and of EurExpress data from March 2006 to November 2007 (in the right figure). Two workloads exhibit a similar structure but different details.

Taking the EurExpress workload as example, when using the histogram to illustrate the distribution shape of the workload values, as shown in the top-right figure of Figure 6.5, a skewed Weibull distribution [Bala 06] can be observed. Other figures in Fig-

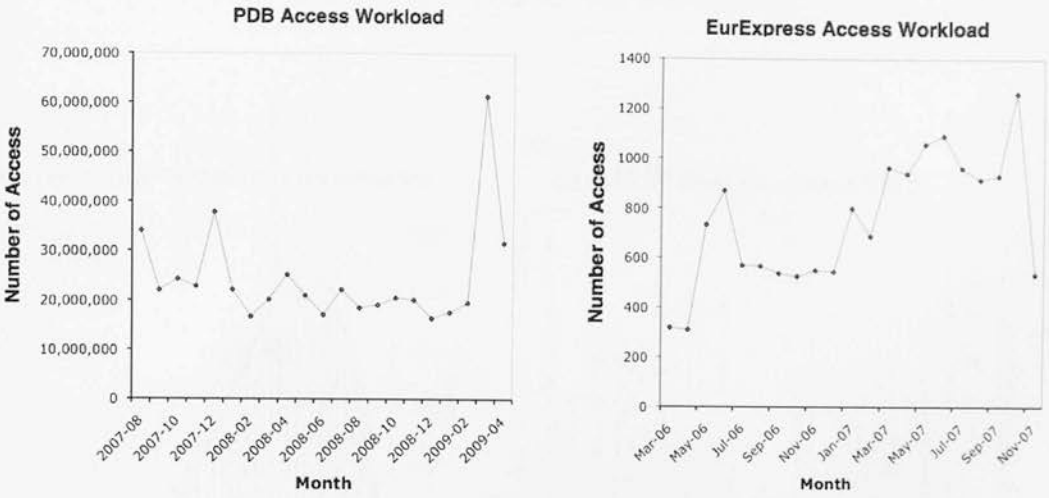


Figure 6.4: PDB and EurExpress access workloads show a normal distribution pattern.

ure 6.5 describe the distribution fitting processes applied to the empirical values. The pdf (probability density function) of Weibull distribution is defined as:

$$f(x) = \frac{\alpha}{\beta}\left(\frac{x}{\beta}\right)^{\alpha-1} e^{-(x/\beta)^{\alpha}} \tag{6.3}$$

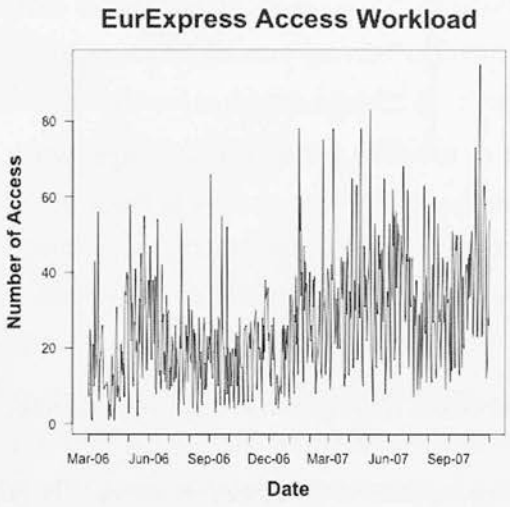
On fitting to the observation data[6], the shape parameter is estimated to be: $\alpha = 1.74$, and the scale parameter is: $\beta = 29.3$.

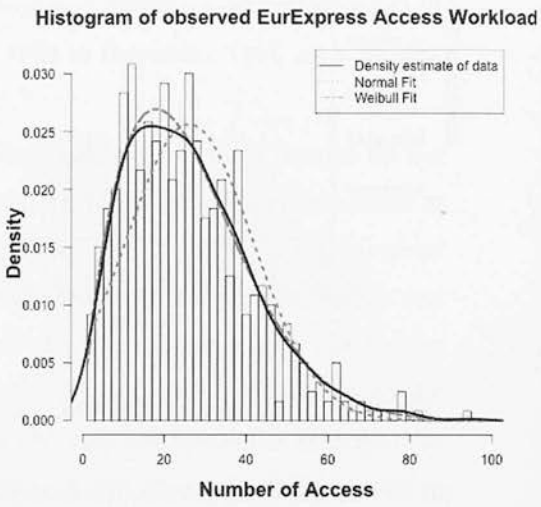
To measure the goodness of fit, the quantile-quantile (Q-Q) plot is used[7], which is shown in the bottom figures in Figure 6.5. Here the quantiles of workload trace and of a sample from the theoretical Weibull distribution are compared. If both sets of data were drawn from the same distribution, a straight line with slope 1 is expected. As shown in the figure, this condition appears to be well satisfied. Accordingly, a synthetic binding-read workload can be modelled by sampling from the distributions that constitute the Weibull distribution features which represent the workload that the system would likely encounter in practice.

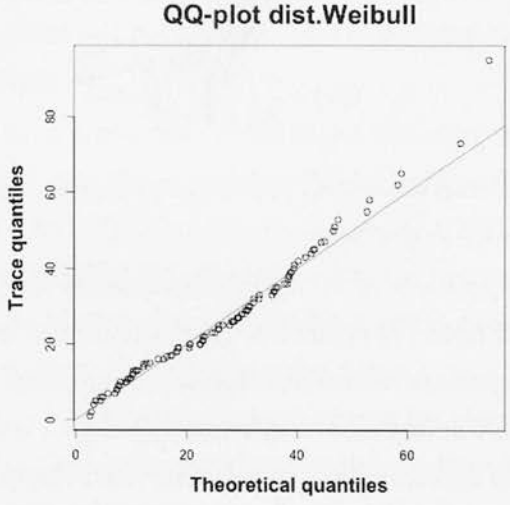
Besides access volume, another workload feature of interest is the relative popularity of data items. Figure 6.6 shows the distribution of the protein structures access frequency in PDB on a log-log scale. The x-axis represents the PDB structure IDs, ordered by descending access frequency. The y-axis refers to the access frequency. The distribution shows the Zipf shape [Bala 06]. 1% of protein structures attracted ~90% access hits.

[6] R statement, fitdistr(), is used to estimate the parameters, which provides a nonlinear fitting method, the maximum-likelihood model.

[7] R statement, qqplot(), is used, which compares the empirical quantiles with the ones from a theoretical Weibull model.







Weibull Fitting Parameters:

Weibull	Scale	Shape
Distribution	29.3306	1.74084
	(0.7247)	(0.0551)

Figure 6.5: Analysis of EurExpress Access Workloads

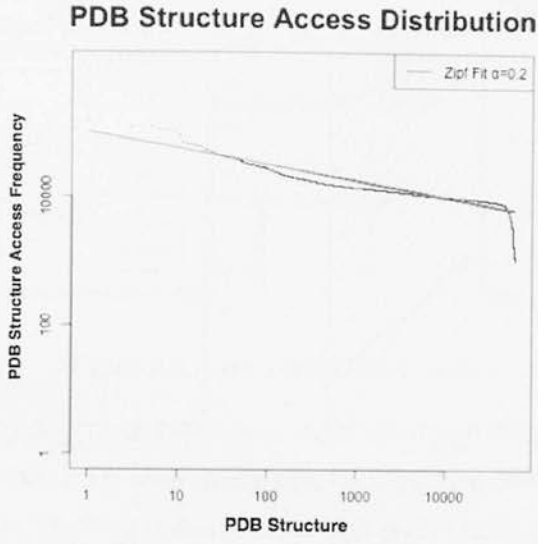


Figure 6.6: Distribution of PDB IDs Access Frequency

This feature suggests optimisation in the design. For example, caching merely 1% of most accessed data items may greatly improve the response efficiency. In contrast, if the accesses are not clustered, the cache strategies may not achieve high performance gains.

Binding Content Workload Patterns

Tagging behaviour is another factor affecting the binding read and write performance. For example, to insert the bindings with more tags will require more computation than the bindings with few attached tags, and to query with frequently used tags will retrieve more bindings than with rarely used tags. The difference could become significant as the binding size increases. [8]

To understand likely tagging behaviour, the data resource of a random snapshot (December 2006) from Flickr is analysed, which includes 3.6 million photos, and 14.9 million tags (among them, 391,601 unique tags). The left figure of Figure 6.7 shows the plot of the tag frequency on a log-log scale. The x-axis represents the unique tags ranking in descending tag frequency. The y-axis refers to the tag frequency. The distribution follows Zipf's law:

$$c \approx C \times i^{-\alpha} \tag{6.4}$$

where C is some constant that reflects the number of samples and the number of items to choose.

The Zipf distribution has been commonly used to model popularity [Adam 02, Li 92]. In this case, the statement indicates the probability of a tag having tag frequency c is

[8] We currently only model the binding tag behaviours. For the other three binding attributes (binding *ID* which is a UUID, binding *subject* and binding *object* which are URIs), we assume the variations of their data size do no have significant effects on binding performance.

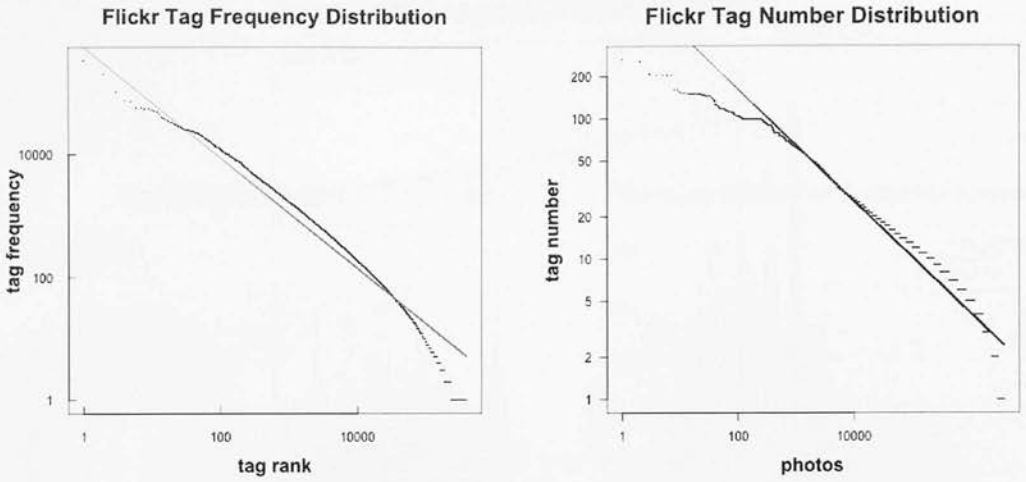


Figure 6.7: Distribution of Flickr Tag Frequency and Tag Number Per Photo

proportional to $i^{-\alpha}$. In other words, being in rank i means that there are i items with access counts larger than c. Similar observations have been reported by other research [Sigu 08, Gold 05], and our results confirm this special tagging behaviours. Fitting the empirical data, the estimated parameter is $\alpha = 0.9$.

The right figure of Figure 6.7 shows the distribution of the number of tags per photo also follows the Zipf distribution. The x-axis represents the 3.6 million photos, ranked by the number of tags per photo in a descending order. The y-axis refers to the number of tags assigned to photo. The probability of having x tags per photo is proportional to $i^{-0.4}$.

The binding workload simulator uses a vocabulary base from WordNet 3.0[9] 144,895 English words, and generates the binding tags in accordance with the tag characteristics in Flickr.

6.1.2 Experimental Setup

The experiment environment is shown in Figure 6.8. A web server is set up to run the binding service. A client consists of a workload simulator and a performance monitor. The simulator bases on the established workload models to generate various workload traces. The monitor collects the performance measures of the binding service in responding the simulated workloads. The measures are stored in a local database, and further delivered to a statistical analysis tool, R[10], which outputs diagrams graphically depicting the results.

The simulator is implemented by using the Stochastic Simulation in Java (SSJ) library[11] developed at the Université de Montreal. The SSJ library provides facilities for generating random variants related to probability distributions, managing a simula-

[9] WordNet: wordnet.princeton.edu

[10] R: http://www.r-project.org/

[11] SSJ: http://www.iro.umontreal.ca/ simardr/ssj/indexe.html

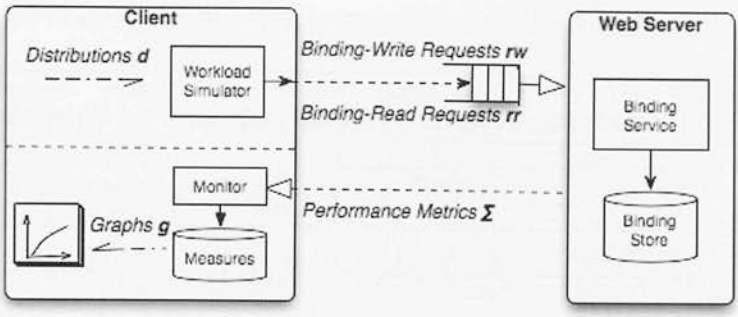


Figure 6.8: Experimental Environment

tion clock, and programming discrete-event simulations with both events and processes [LEcu 02]. Classes from other open source Java libraries are used, including the Colt, developed at the CERN. The Colt is designed for high performance scientific and technical computing in Java. It provides the efficiency and convenience of the statistics implementation [Hosc 04]. These classes are used to implement special distributions such as Zipf which is absent from most of the Java mathematics libraries.

The machine types are as follows: Intel(R) Core(TM)2 Quad CPU Q9400 2.66GHz, RAM 7GB, Hard Disk 144GB, 100Mbps network interface, OS Red Hat 4.1.2-42. The web server is configured with Tomcat 5.5, OGSA-DAI server settings, and MySQL 6.0. The client is equipped with Java 1.5, R 2.9, and MySQL database 6.0.

A benchmark is run to control the experiment workflow, which consists of a set of command scripts that clean up the storage, starts the simulator to generate binding read or write workloads, collects and stores the measures, and delivers the measurements to R. Each configuration is repeated 10 times, and the mean values, the Standard Errors (SEs) and the 95% Confidence Intervals (95% CIs) are calculated.

Experimental Configurations

Table 6.1 lists the distribution models and the input parameters used in the experiments. It serves as a reference table. The contents will be explained when we describe the simulation details. Briefly, the two main variables are the workload type and the scale of request. The workload types are determined by the input distribution models constructed from the real-world observations. The scales of the requests are small, moderate, large, and ultra.

In the case of binding-write workloads, the *Large Scale* in Table 6.1 is configured with the parameters obtained from the observed data, since the studied systems are moderate to large repositories. From here, the *Moderate Scale* is devised, which is roughly 5 times smaller, and the *Small Scale* which is roughly 10 times smaller. The *Ultra Scale* is intended for the limit test. With current system settings, it is configured as roughly 2 times larger than the *Large Scale*. Table 6.2 lists the actual generated data volumes for each scale.

Table 6.1: Distribution Models and Parameters Used in the Experiments

Process	Model	Small Scale	Moderate Scale	Large Scale	Ultra Scale
Poisson Process	$\mathcal{A}$: Exponential Dist.				
Arrival Interval	$f(x) = \lambda e^{-\lambda x}, x \geq 0$	$\lambda_{Burst} = 0.02$	$\lambda_{Write} = 1.0$	$\lambda_{Read} = 2.0$	$\lambda_{Read} = 10.0$
$\mathcal{A}$					
Binding Write	Uniform Workload	$a = 1$	$a = 6$	$a = 12$	$a = 30$
Request Size	$\mathcal{N}_{Write}$: Uniform Dist.	$b = 80$	$b = 400$	$b = 888$	$b = 1600$
$\mathcal{N}_{Write}$	$f(x) = 1/(b-a)$,				
	$a \leq x \leq b$				
	Growth Workload	$A_0 = 10$	$A_0 = 10$	$A_0 = 10$	$A_0 = 10$
	$\mathcal{N}_{Write}$:Polynomial Dist.	$A = 0.5$	$A = 2.5$	$A = 5$	$A = 10$
	$f(t) = A_0 + A \times t + \xi_t$				
	ξ: Uniform Dist.	$a_\xi = -10$	$a_\xi = -10$	$a_\xi = -10$	$a_\xi = -10$
	$f(\xi) = 1/(b_\xi - a_\xi)$,	$b_\xi = 10$	$b_\xi = 10$	$b_\xi = 10$	$b_\xi = 10$
	$a_\xi \leq \xi \leq b_\xi$				
	Burst Workload Pattern	$\lambda_{Background} = 1.0$	$\lambda_{Background} = 1.0$	$\lambda_{Background} = 1.0$	$\lambda_{Background} = 1.0$
	Arrival Interval $\mathcal{A}_{Background}$,	$\lambda_{Burst} = 0.02$	$\lambda_{Burst} = 0.02$	$\lambda_{Burst} = 0.02$	$\lambda_{Burst} = 0.02$
	$\mathcal{A}_{Burst}$:Exponential Dist.				
	$f(x) = \lambda e^{-\lambda x}, x \geq 0$	$a_{Background} = 1$	$a_{Background} = 5$	$a_{Background} = 10$	$a_{Background} = 30$
	Request Size $\mathcal{N}_{Background}$,	$b_{Background} = 80$	$b_{Background} = 400$	$b_{Background} = 800$	$b_{Background} = 1600$
	$\mathcal{N}_{Burst}$: Uniform Dist.				
	$f(x) = 1/(b-a)$,	$a_{burst} = 10$	$a_{Burst} = 50$	$a_{Burst} = 100$	$a_{Burst} = 300$
	$a \leq x \leq b$	$b_{Burst} = 800$	$b_{Burst} = 4000$	$b_{Burst} = 8000$	$b_{Burst} = 16000$
Binding Write	$\mathcal{F}$:Zipf Dist.				
Tag Frequency	$f(x, \alpha, N) = \frac{1/x^\alpha}{\sum_n^N (1/n^\alpha)}$	$\alpha = 0.9$	$\alpha = 0.9$	$\alpha = 0.9$	$\alpha = 0.9$
Distribution $\mathcal{F}$					
Binding Write	$\mathcal{L}$:Zipf Dist.				
Tag Length	$f(x, \alpha, N) = \frac{1/x^\alpha}{\sum_n^N (1/n^\alpha)}$	$\alpha = 0.4$	$\alpha = 0.4$	$\alpha = 0.4$	$\alpha = 0.4$
Distribution $\mathcal{L}$					
Binding Read	$\mathcal{L}$:Exponential Dist.				
Arrive Interval:	$f(x) = \lambda_{Read} e^{-\lambda_{Read} x}$,	$\lambda_{Read} = 10.0$	$\lambda_{Read} = 10.0$	$\lambda_{Read} = 10.0$	
$\mathcal{A}_{Read}$	$x \geq 0$	$\alpha = 1.74$	$\alpha = 1.74$	$\alpha = 1.74$	—
Request Size	$\mathcal{N}_{Read}$:Weibull Dist.	β=1/15.0	β=1/29.3	β=1/58.6	
$\mathcal{N}_{Read}$	$f(x) = \frac{\alpha}{\beta}\left(\frac{x}{\beta}\right)^{\alpha-1} e^{-(x/\beta)^\alpha}$				
Combined	$\mathcal{A}$:Exponential Dist.	$\lambda_{Write} = 1.0$	$\lambda_{Write} = 1.0$	$\lambda_{Write} = 1.0$	
Workloads	$f(x) = \lambda e^{-\lambda x}, x \geq 0$	$\lambda_{Burst} = 0.02$	$\lambda_{Burst} = 0.02$	$\lambda_{Burst} = 0.02$	
Arrive Interval:		$\lambda = 2.0$	$\lambda = 2.0$	$\lambda = 2.0$	
$\mathcal{A}_{Write}, \mathcal{A}_{Burst}$,	$\mathcal{N}_{Write}$: See above				—
$\mathcal{A}_{Read}$	$\mathcal{N}_{Read}$: Weibull Dist.	$\alpha_{Read} = 1.74$	$\alpha_{Read} = 1.74$	$\alpha_{Read} = 1.74$	
Request Size:	$f(x) = \frac{\alpha}{\beta}\left(\frac{x}{\beta}\right)^{\alpha-1} e^{-(x/\beta)^\alpha}$	$\beta_{Read} = 1/15.0$	$\beta_{Read} = 1/29.3$	$\beta_{Read} = 1/58.6$	
$\mathcal{N}_{Write}, \mathcal{N}_{Write}$					

Note: For Binding Write: *SmallScale*:1×time, *ModerateScale*:5×times; *LargeScale*:10×times; *UltraScale*: 20×times. For Binding Read: *Small Scale*:1×time, *Moderate Scale*:2×times; *Large Scale*:4×times.

Table 6.2: Statistics of Binding Scales in Simulations

Workload Patterns	Scale	Total Number of Bindings	Total Number of Tags	Number of Unique Tags	Data Size (MB)
Write (Uniform)	*Small*	4,622	180,348	53,510	18.42
	Moderate	23,132	475,979	91,335	50.27
	Large	46,155	720,287	118,476	79.52
	Ultra	92,814	1096,293	123,628	129.38
Write (Growth)	*Small*	3,859	161,693	49,996	16.52
	Moderate	14,947	366,205	80,284	38.01
	Large	34,324	602,735	101,220	65.18
	Ultra	56,495	813,669	113,206	91.38
Write (Burst)	*Small*	5,236	194,287	56,116	19.87
	Moderate	26,175	512,748	94,526	54,52
	Large	52,355	776,929	111,404	86.66
	Ultra	105,364	1,283,028	126,109	141.79

Note: the statistics are collected after each workload simulation running for 100 simulated hours.

In the case of binding-read workloads, the *Moderate Scale* is set up as the baseline using the parameters obtained from the observation data, and the *Large Scale* is configured as 2 times larger, and the *Small Scale* is 2 times smaller, which are sufficient to distinguish them in the experiments.

Experimental Verification of the Simulated Workloads

To ensure that the simulator reliably generates the desired workload traces, the outputs of each simulation attribute are verified, and presented below.

(1) Simulation of Arrival Interval, $\mathcal{A}$

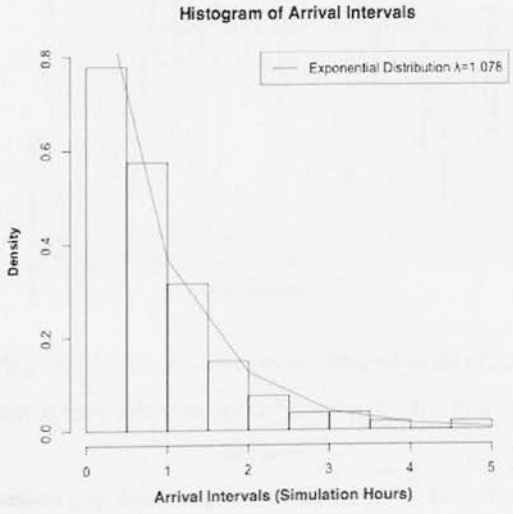


Figure 6.9: The simulated request-arrival intervals present an Exponential Distribution $f(x) = \lambda e^{-\lambda x}, x \geq 0$, where $\lambda = 1.078$, satisfying a Poisson Process.

A very common assumption for the arrival process is that requests arrive in a Poisson process [Gros 98]. A Poisson process is a stochastic process in which events occur continuously and independently of one another. Examples that are well-modelled as Poisson processes include the concurrent telephone calls arriving at a switchboard, and page view requests to a website. In a Poisson process, the inter-arrival times are exponentially distributed. Consider a period of time $\mathcal{T}$, during which events occur at an average rate of λ requests per time unit. A fixed time horizon of $\mathcal{T} = 100$ simulated hours is used, and Figure 6.9 shows the histogram of the request arrival intervals generated by the simulator, with the fitting line of an Exponential Distribution, $f(x) = \lambda e^{-\lambda x}, x \geq 0$, where $\lambda = 1.078$. This is approximately the same as the configuration input 1.0.

(2) Simulation of Binding Write Volume, $\mathcal{N}_{Write}$

When simulating the binding-write workloads, each Poisson arrival event triggers $\mathcal{N}_{Write}$ number of binding creations[12], where $\mathcal{N}_{Write}$ is determined by one of the three investigated workload patterns, *Uniform*, *Continuing Growth*, or *Burst*.

A. Uniform. Figure 6.10 is the simulation results of the uniform workload pattern. During $\mathcal{T} = 100$ simulated hours, the binding-write requests arrive randomly. Each request creates $\mathcal{N}_{Write}$ bindings, where $\mathcal{N}_{Write}$ is produced by a Uniform Distribution, $f(x) = 1/(b-a), a \leq x \leq b$, with $a = 1$ and $b = 80$. (The value of a and b are the distribution fittings of the annotation workload of the EurExpress.) The simulated workload shows the desired Uniform pattern.

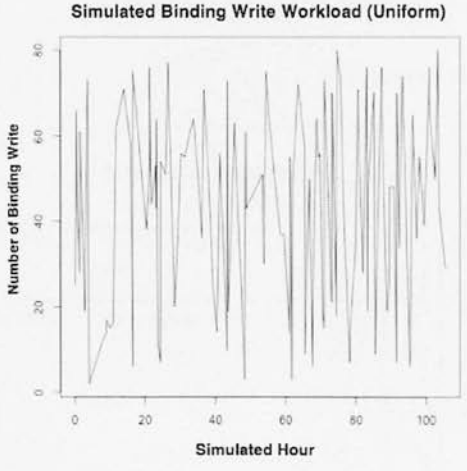


Figure 6.10: The simulated binding-write workload (Uniform) using the *Small Scale* configuration. The x-axis refers to the simulated hours, and the y-axis refers to the numbers of binding-write per request.

[12]The simulator generates binding-write requests invoking the binding-creation operation which is one the important binding-write behaviours. Other types of binding-write behaviours, including update of binding tags, and updates of binding *subject*/*object*, are not simulated, which could be included in future work when relevant characteristic workload observations are available.

B. Continuing Growth. To simulate the Continuing Growth workload, during $\mathcal{T} = 100$ simulated hours, each randomly arrived binding-write request creates $\mathcal{N}_{Write}$ bindings, where $\mathcal{N}_{Write}$ is computed from the Polynomial distribution, $f(t) = A_0 + A \times t + \xi$, for $A_0 = 10$, $A = 0.5$, (the value of A_0 and A are the distribution fitting of PDB data generation workload.) ξ is generated from a Uniform Distribution, $f(\xi) = 1/(b_\xi - a_\xi), a_\xi \leq \xi \leq b_\xi$, where $a_\xi = -10$ and $b_\xi = 10$. Figure 6.11 gives the results showing that simulated workload exhibits the intended Continuing Growth characters.

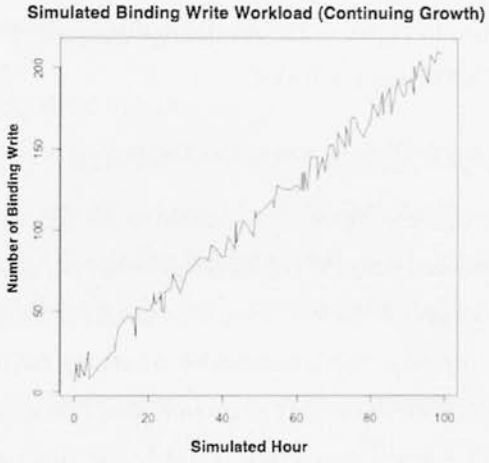


Figure 6.11: The simulated binding-write workload (Continuing Growth) using the *Small Scale* configuration. The x-axis refers to the simulated hours, and the y-axis refers to the numbers of binding-write per request.

C. Burst. Figure 6.12 presents the simulated workload comprising bursts. To model the bursts, a Hidden Markov Model (HMM) is constructed, consisting of two-state Markov

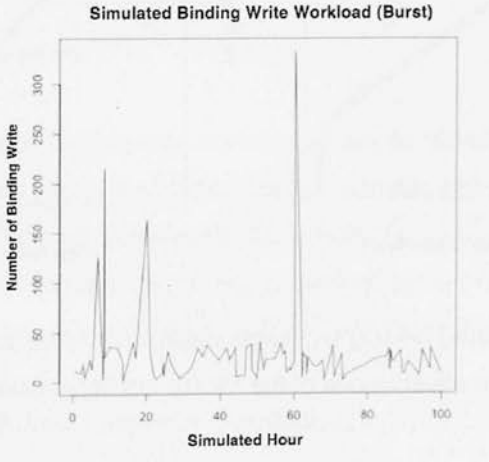


Figure 6.12: The simulated binding-write workload (Burst), using the *Small Scale* configuration. The x-axis refers to the simulated hours, and the y-axis refers to the numbers of binding-write per request.

chains which are two independent Poisson processes. The arrival intervals for the background updates are generated from an Exponential Distribution, $f(x) = \lambda_{Background} e^{-\lambda_{Background} x}$, $x \geq 0$, with $\lambda_{Background} = 1.0$. The requests sizes are calculated from a Uniform Distribution, $f(x) = 1/(b_{Background} - a_{Background}), a_{Background} \leq x \leq b_{Background}$, with $a_{Background} = 1$ and $b_{Background} = 80$. For the bursts, the arrival intervals are generated from a second Exponential Distribution, $f(x) = \lambda_{Burst} e^{-\lambda_{Burst} x}, x \geq 0$, using $\lambda_{Burst} = 0.02$, and the number of requests are generated from a Uniform Distribution, $f(x) = 1/(b_{Burst} - a_{Burst}), a_{Burst} \leq x \leq b_{Burst}$, with $a_{Burst} = 10$ and $b_{Burst} = 800$, which means the scale of burst is 10 times that of background updates. As shown in Figure 6.12, the simulation result is very close to the observations from the NanoCMOS and the BADC.

(3) Simulation of Tag Length Distribution $\mathcal{L}$ and Tag Frequency $\mathcal{F}$.

Two attributes related to the binding-content simulation are the tag length distribution and the tag frequency. When simulating the tag length distribution, each binding-write request is controlled to choose n words from WordNet, then invoke the binding service operation, CreateBinding(ref_sub, ref_obj, $tags$), to generate a binding having n tags attached[13]. n is produced by a Zipf distribution number generator which computes the Zipf probability mass function $f(x, \alpha, N) = \frac{1/x^{\alpha}}{\sum_{n=1}^{N}(1/n^{\alpha})}$, using $\alpha = 0.4$. As depicted in the left figure of Figure 6.13, the simulated tag length distribution presents the same Zipf shape as that in the Flickr.

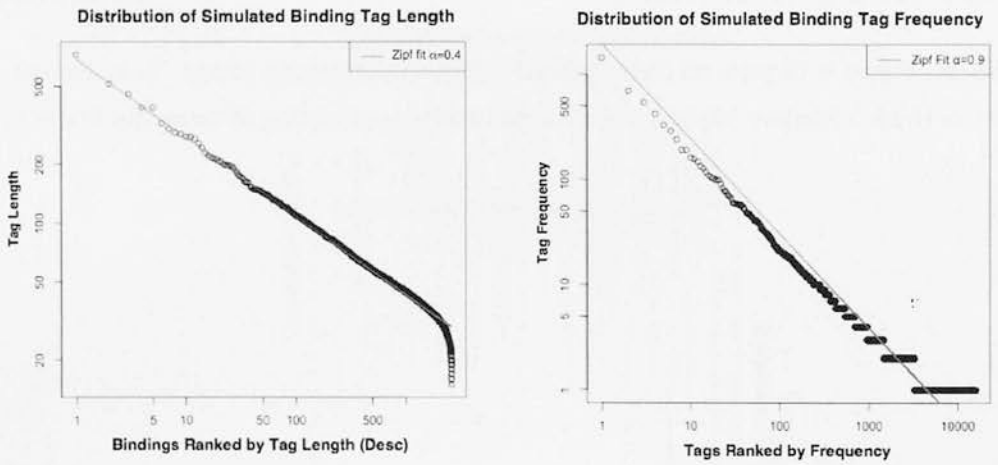


Figure 6.13: The simulated binding tag length distributions (left) and tag frequency distributions (right) match the observations of the Flickr's tag behaviours, presenting Zipf distributions.

[13]When the simulator invokes CreateBinding(ref_sub, ref_obj, $tags$) the other two parameters are set as follows: ref_sub = "http://binding.data.k", ref_obj = "http://binding.metadata.k", where k is a counter increasing for each new binding.

To model the tag frequency, the selection of tags is determined by a second Zipf distribution, ($f(x,\alpha,N) = \frac{1/x^{\alpha}}{\sum_{n=1}^{N}(1/n^{\alpha})}$, for $\alpha = 0.9$). When selecting a tag, the simulator reads a random numbers from the Zipf's number generator, and maps it to a tag id in WordNet, thus the desired tag can be located, which is one of the tags used by CreateBinding(ref_sub, ref_obj, $tags$) to create the binding. In the same way, other tags are determined. Since the tag lists are handled as *Set*, repeated tags are not allowed. Any duplicated tag for the same binding has to be removed, and re-selected until a distinguished tag appears. Illustrated by the right figure of Figure 6.13, the simulated tag frequency captures the Flickr's tag behaviours.

(4) Simulation of Binding Read Volume, $\mathcal{N}_{Read}$

On simulating the binding-read workload, each Poisson arrival event triggers $\mathcal{N}_{Read}$ binding reads, where $\mathcal{N}_{Read}$ is generated from a Weibull Distribution. Figure 6.14 depicts the simulated binding-read workload. The histogram illustrates that the Weibull-shape distribution of $\mathcal{N}_{Read}$ approximates the real-world observations.

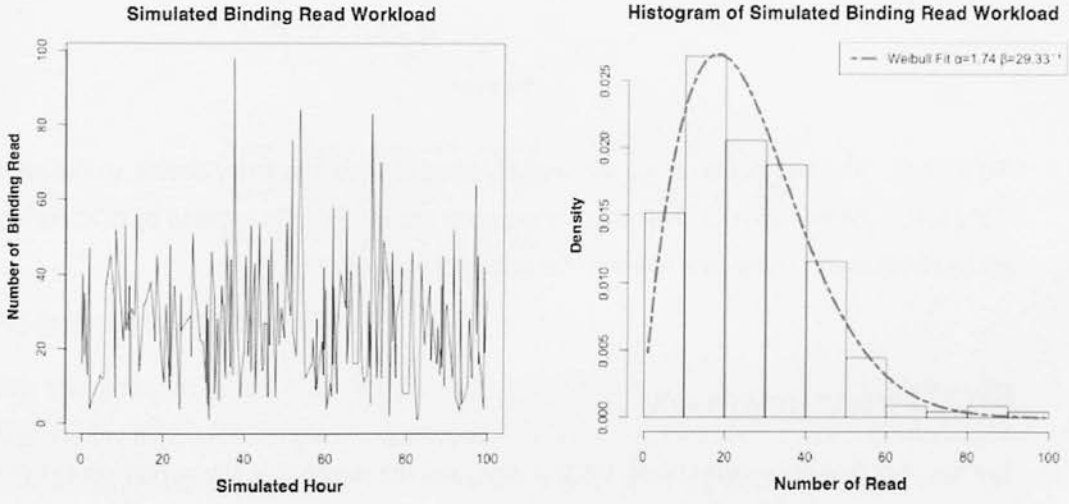


Figure 6.14: The simulated binding-read workload shows an intended Weibull Distribution. During $\mathcal{T} = 100$ simulated hours, the binding access requests arrive randomly. The arrival intervals are generated from an Exponential Distribution, $f(x) = \lambda_{Read} e^{-\lambda_{Read} x}$, with $\lambda_{Read} = 1.0$. Each binding access request triggers $\mathcal{N}_{Read}$ binding reads, where $\mathcal{N}_{Read}$ is generated from a Weibull Distribution, $f(x) = \frac{\alpha}{\beta}\left(\frac{x}{\beta}\right)^{\alpha-1} e^{-(x/\beta)^{\alpha}}$, with $\alpha = 1.74$, $\beta = 29.33^{-1}$.

(5) Simulated Binding Access Frequency Distribution, $\mathcal{P}$

The final attribute that needs to be examined is the bindings access should follow the popularity distribution identified in the PDB. This is simulated as follows: each time the simulator obtains a random number from a Zipf distribution number generator which is

configured with a probability mass function $f(x,\alpha,N) = \frac{1/x^{\alpha}}{\sum_{n=1}^{N}(1/n^{\alpha})}$ for the given parameters $\alpha = 0.2$, and N is the number of bindings in the store. (Note, the value of α comes from the distribution fitting of the workloads of the PDB.) The simulator then maps the random number to a binding record identifier[14] and obtains the record value. The obtained value is delivered as the query parameter to the binding service discovery operation[15] to retrieve the binding. The simulated binding access popularity is shown in Figure 6.15, and the intended Zipf shape can be observed.

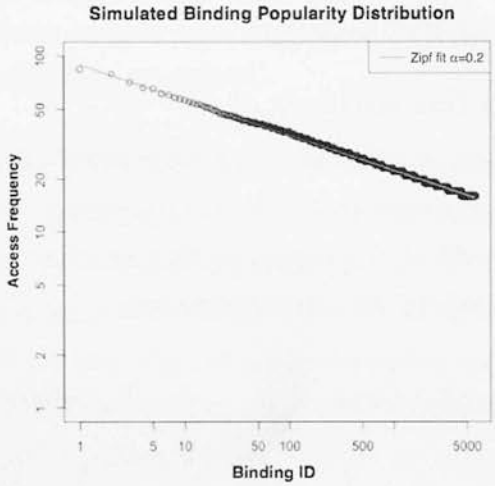


Figure 6.15: The distribution of the simulated binding access frequency shows an intended Zipf distribution with $\alpha = 0.2$. The x-axis represents the Binding IDs ordered by descending access frequency. The y-axis refers to the access frequency.

6.1.3 Experimental Results

Having prepared the environment, the test data, and the workloads, the experiment is run to collect the measures.

Two main measures, collected under different workloads and varied request sizes, are the responsiveness and productivity of the binding service. Responsiveness is the evaluation of the speed with which a service can process a request. One of the common measures is response time, which is defined as the average amount of time (in seconds) required for a service component to accomplish a given task, and is measured from the time a requester initiates the invocation to the time the requester receives the last byte of the response [Jain 91]. Productivity is the evaluation of the number of requests a service

[14] For the simulation purpose, each binding record has an index number as the record identifier.

[15] The current simulator uses operation GetBindingByID() for binding access. It guarantees that the performance measures of binding-read are taken under the same size of binding retrievals (thus the results are comparable). Each binding-read retrieves n bindings, $n = 1$.

processes. One of the major measures is throughput, which is measured in units of work accomplished per unit time [Jain 91]. Other performance metrics, such as reliability, security, availability, utilisation and cost are application dependent, and therefore are not included at the stage of concept proving.

Note, two time systems are used in the experiments. Each configured experiment runs 50 simulation hours, and performance of binding service are measured in Wall Clock Time (WCT)[16]. Table 6.3 lists the actual WCT in seconds each experiment takes to run 50 simulation hours in testing small and large scale workloads. Using simulation method reduces the experiment time considerably.

Table 6.3: Wall Clock Time of Experiments

Workload	Types	Small Scale (*sec*)	Large Scale (*sec*)
Creation	*Uniform*	132	980
	Growth	141	289
	Burst	196	1174
Access		124	437
Combined	*Uniform+Access*	132	872
	Growth+Access	231	368
	Burst+Access	221	1256

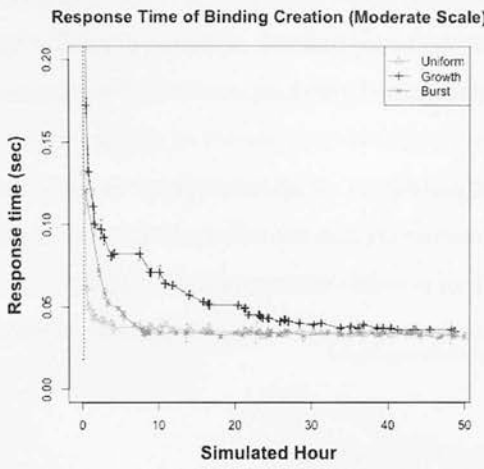
Note: Each configured experiment runs 50 simulation hours. The table gives the WCT in seconds for testing small and large scale workloads.

Performance of Binding Write

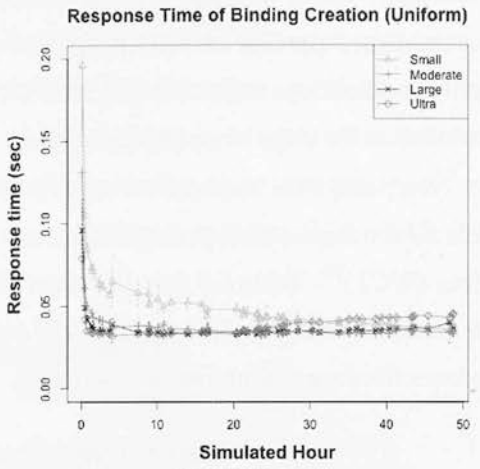
The first group of experiments is aimed to evaluate the performance of binding write. The binding-creation performance under 3 different type and 4 different scales of workloads are measured. Each configuration is run ten times, means and 95% Confidence Intervals are collected, and a total 120 trials are attempted.

Figure 6.16 presents the service response time. The left three graphs (a)(c)(e) group the measures by scales and compare the service performance in handling different types of workloads. The same data are reproduced in the right three graphs (b)(d)(f), which group the measures by workload types and compare the service performance in handling different scales of workloads. In each figure, the x-axis represents the simulated time in hours, and the y-axis refers to the response time in (WCT) seconds. Thus, each point represents the average response time (for a single binding-creation) of the binding service in handling a binding-creation request arrived at the corresponded simulation-time.

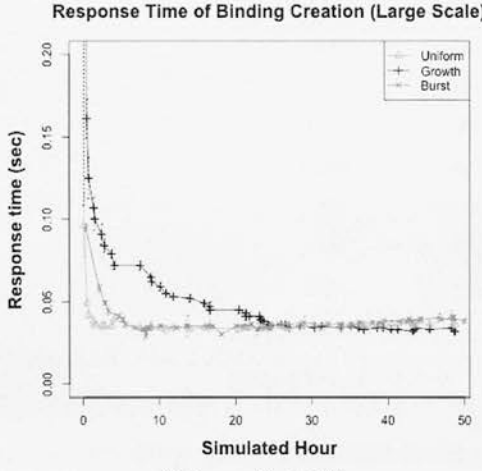
[16]Wall Clock Time is the actual time taken by a computer to complete a task. It is the sum of three terms: CPU time, I/O time, and the communication channel delay.



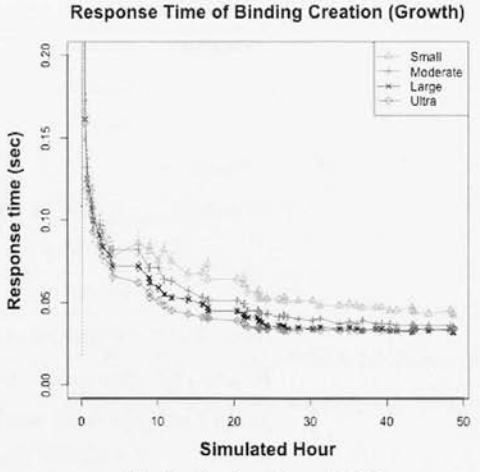
(a) Moderate Scale WL



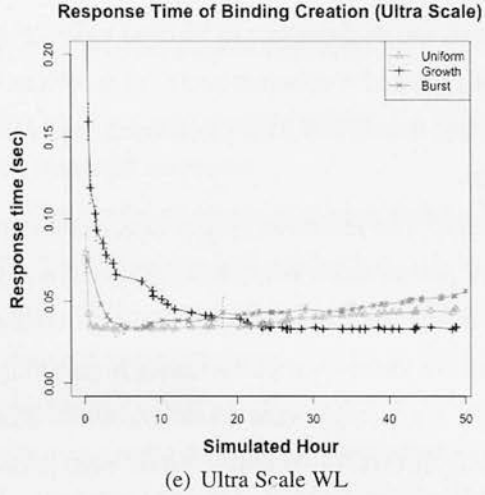
(b) Uniform WL



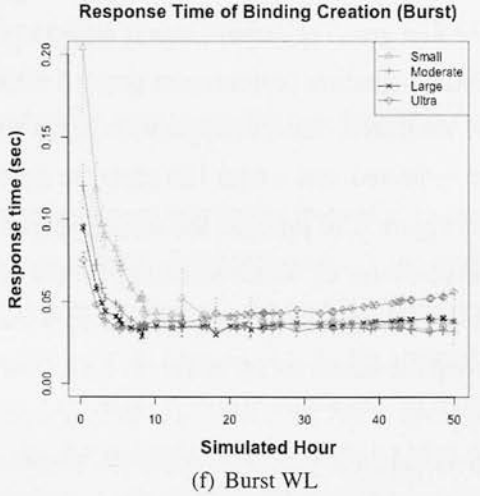
(c) Large Scale WL



(d) Continuing Growth WL

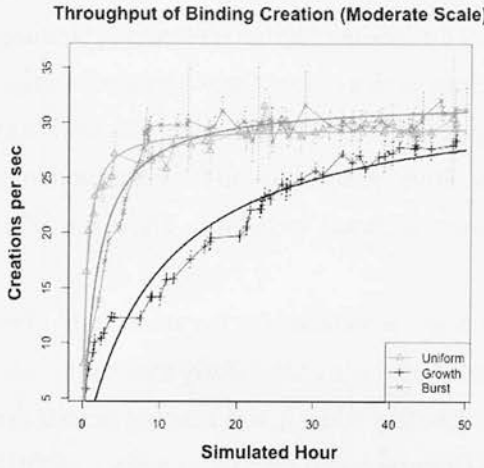


(e) Ultra Scale WL

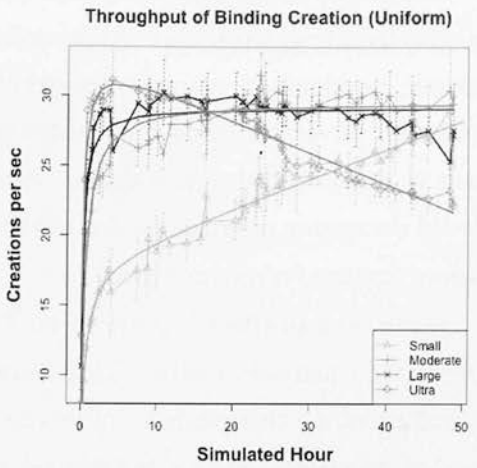


(f) Burst WL

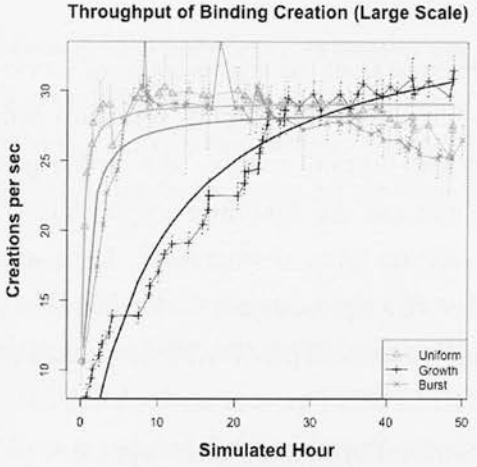
Figure 6.16: Response time of binding creation. The x-axis represents the simulated time in hours. The y-axis represents the binding-creation response time in (WCT) seconds. Each data point is the mean of 10 trials, and error bars show the Confidence Intervals (*CIs*), approximately *Mean* ± *2* × *Standard Error* [Cumm 07], which gives a range of values with 95% confident contains the true mean.



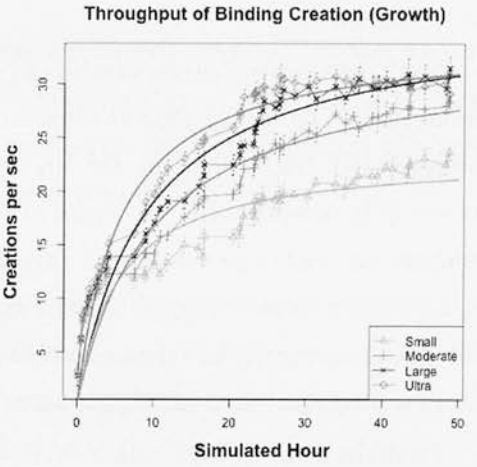
(a) Moderate Scale WL



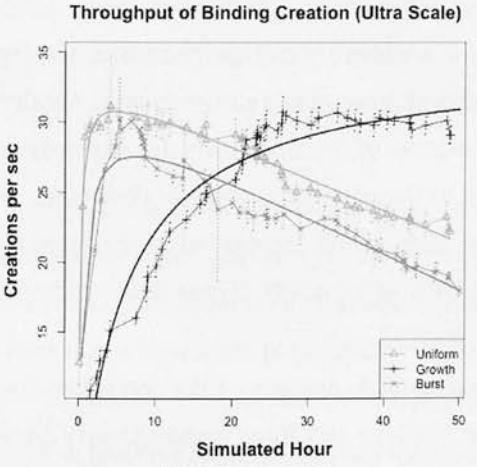
(b) Uniform WL



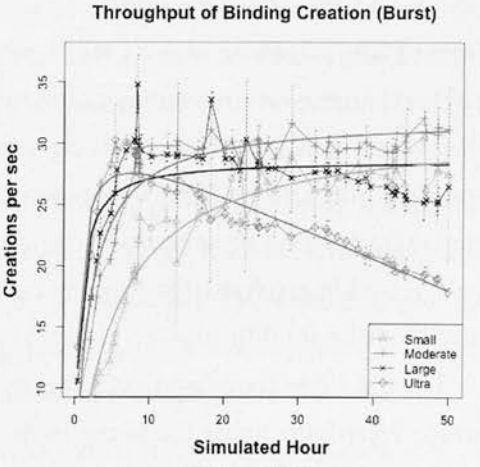
(c) Large Scale WL



(d) Continuing Growth WL



(e) Ultra Scale WL



(f) Burst WL

Figure 6.17: Throughputs of binding creation, associated with regression lines. The x-axis represents the simulated time in hours. The y-axis represents the number of binding-creation per (WCT) second. Each data point is the mean of 10 trials, and error bars show the Confidence Intervals (*CIs*), approximately *Mean* ± *2* × *Standard Error*, which gives a range of values with 95% confident contains the true mean.

The graphs illustrate that the three types of workloads are handled very similar for small to large scale workloads, which implies that the binding service is robust for binding-creation workloads of various types and of various scales (when the scales are manageable). All curves start from a high value and quickly decrease. The larger the workload scale is, the quicker the curve decays. After the initial jumps, the curves either continue slowly decreasing or remain at that level without significant variations. These are satisfactory features for response time.

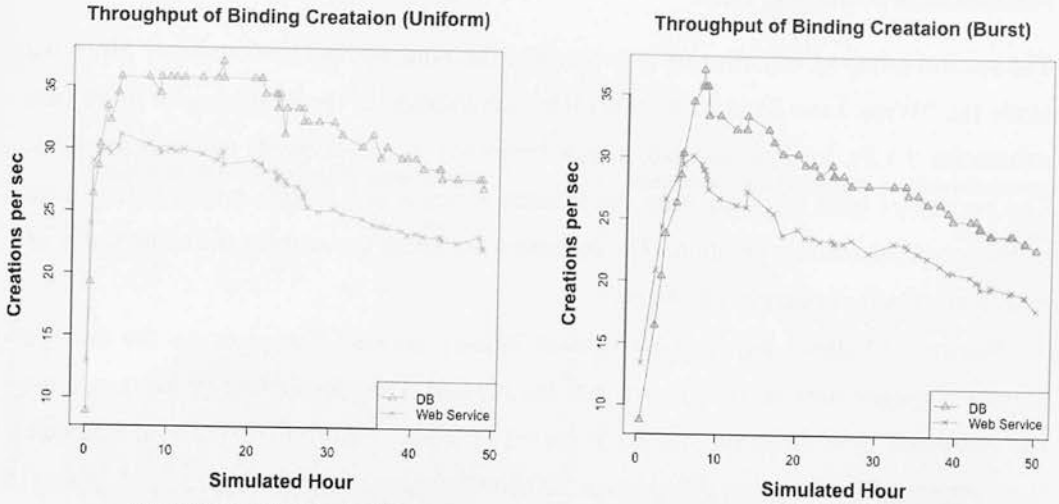
It has been an effort to construct the *Ultra Scale* workloads for the stress tests. The graphs show that the simulated *Ultra Scale* workloads have successfully enabled the observations of the characteristics of service performance when it was running near/at the limit of the system capacity. In graph (e), both *Uniform* and *Burst* curves show a slightly increasing trend, in other words, the binding service responds slower and slower (and does not get back).

More performance characteristics can be observed from the visualisations of service throughputs. Figure 6.17 shows the measures of the service throughputs, associated with the regression lines[17]. Again, the left three graphs (a)(c)(e) compare the service performance in handling different scales of workloads, and the right three graphs (b)(d)(f) compare the service performance in handling different types of workloads. The x-axis represents the simulated hours of measures, the y-axis represents the throughput. Thus, each point represents the average number of binding-creation per (WCT) second against the simulated time when the request occurred.

From the small to large scale workloads, the curves of three different types of workloads present similar trends. The throughputs begin from low values and increasing quickly. Afterwords, they either continue increasing or remain at relevant steady level. From (b)(d)(f) it can be seen for the same type of workload, the larger the scales are, the quicker the curves can reach the steady level, and the higher values can be gained. Another observation is that the throughputs capture the aspects of the workloads, for example, in graph (d), the growing-trend shapes correspond to the continuing growth workloads. In graph (f), there are some peaks of the creation rates which correspond to the burst requests. And in graph (b), the *Uniform* curves look relative steady. These are satisfactory characteristics for throughputs.

For the *Ultra Scale* workloads, graph (e) gives more pictures of the service performance behaviours under the heavy loads. The *Uniform* and *Burst* curves clearly show decreasing-trends. After reaching the top points, both curves begin to drop and end up even lower than the smaller scale workloads, which implies the system was running in a very inefficient mode.

[17] Using non-linear regression model, fitted with function, $y = \frac{a \times x}{b+x} - c \times x$, by applying R nls() statement.



(a) Uniform Ultra Scale WL

(b) Burst Ultra Scale WL

Figure 6.18: Performance of binding web service vs. binding database in handling Ultra Scale workloads. The plots are the numbers of binding-creation per (WCT) second (y-axis) against simulated time in hours (x-axis). The upper curves are the throughputs of binding database, the lower curves are the throughputs of binding web service.

To find out the reasons for this phenomena, we check the system usage and processor activities, and notice that CPU is saturated (while the memory and disk are not full). The (MySQL) database uses up >80% CPU, and the web application uses up >15% CPU. It can be concluded that the database causes the problem.

To investigate the problem, we isolate the database from the binding service and examine the binding system performance without the overheads of the web service. The simulator is modified, and the scripts of invoking binding web service are interpreted to directly access binding database. *Ultra Scale Uniform* and *Burst* workloads are used for testing.

Figure 6.18 shows the results and compares the binding performance with and without web service layer. Graph (a) compares the throughputs under *Uniform* type workloads, and graph (b) compares the throughputs under *Burst* type workloads. In both graphs, the upper curve shows the performance of binding database, and the lower curve shows the performance of binding service (binding database + web service layer).

The graphs illustrate with and without the web service layer, the throughputs of binding database present similar decreasing trends, indicating the binding database, which is built on a single server node, becomes saturated under the heavy workloads.

We will come back to discuss the solutions to scale up the binding storage in section 5.4. Before that, let us continue to look at results of other feasibility tests.

Performance of Binding Read

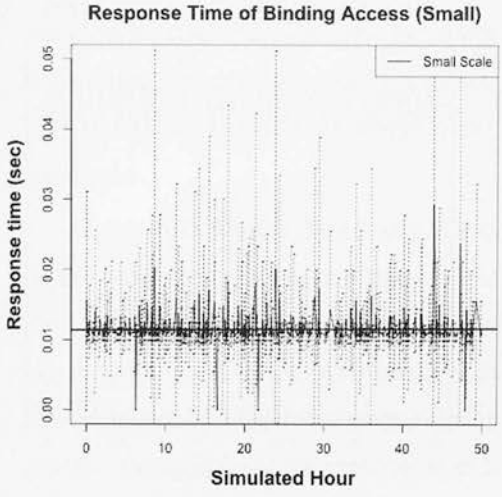
The second group of experiments is to measure the binding-read performance. Since we made the "Write Less Read More" (WLRM) assumption at the beginning of work (see subsection 5.1.3), for binding-read, a high-frequency access-requests rate is configured. The parameter used for simulating the Poisson Process is $\lambda = 10.0$, which is ten times of that used for binding creation. The measures are taken under three different scales of workload: small, moderate, and large.

Figure 6.19 shows the response time of binding access. The plots are the average service response-time in (WCT) seconds for reading a single binding (y-axis), against the simulated time when the request occurred (x-axis). Figure 6.20 shows the service throughputs. The plots are the average numbers of access per (WCT) second (y-axis) against the simulated time when the request occurred (x-axis). In both figures, graphs (a) (b)(c) are the service performance of handling individual access workload. These data are reproduced and displayed together in (d) for comparisons.

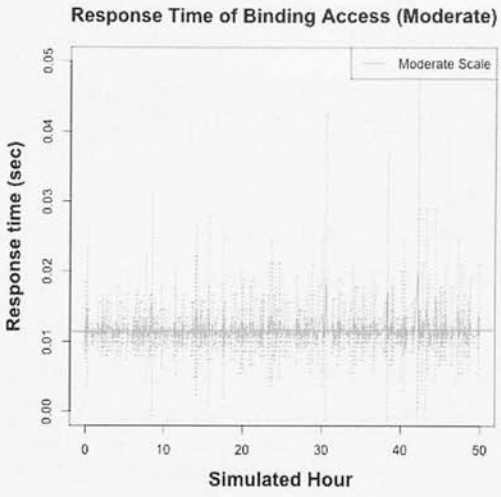
For both response time and throughput, the shapes of the curves are very similar for all three scales of workloads, and the regression lines[18] of the measures (almost) overlap in graph (d). Recall the *Large Scale* access workload is indeed 4 times larger than the *Small Scale* workload in term of numbers of binding accessed, nevertheless, the scale variation appears to have little effect on the service response time and throughputs. On the other hand, the performance differences of dealing with varied scales of binding-creation workloads have been identifiable. These suggest that binding-access performance tends to be more consistent than the binding-creation performance, which implies that binding behaviour is more predictable in dealing with access than creation.

Another observation is that binding-access performance is better than binding-creation performance, i.e., the average binding-access response time is 0.012 (WCT) seconds, comparing to which, the best binding-creation response time is 0.027 (WCT) seconds (more than 2 times slower). The issue relates to the system design. The binding creation operation (CreateBinding()) involves a set of activities and database updates, including: an insert of binding into BindingTable, inserts of (new) tags into TagTable, inserts of binding-tag relationships into BindingTagTable, and updates of the table indexes. The design gives overheads for binding-creation in order to facilitate binding-access, i.e., indexes improve the speed of data retrieval on tables at the cost of slower writes. Using TagTable to store binding tags can ease tag-based queries (and statistical analysis), but has overheads of updating multiple tables. The empirical results confirm the performance gains in binding-read, which implies that the binding service is capable of supporting 'Write Less Read More' scenarios.

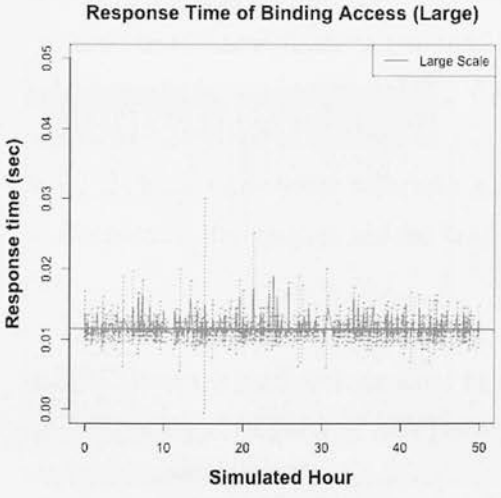
[18] Using linear regression by applying R lm() statement.



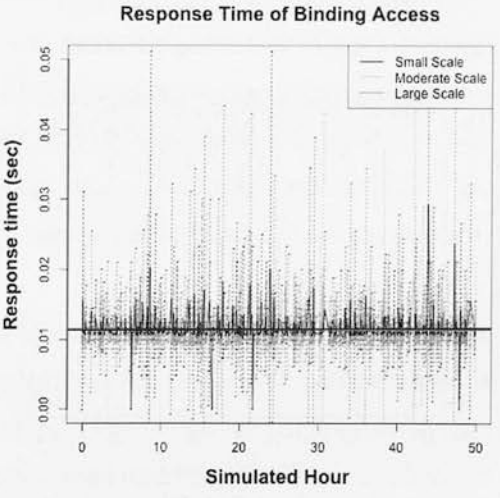
(a) Small Scale WL



(b) Moderate Scale WL

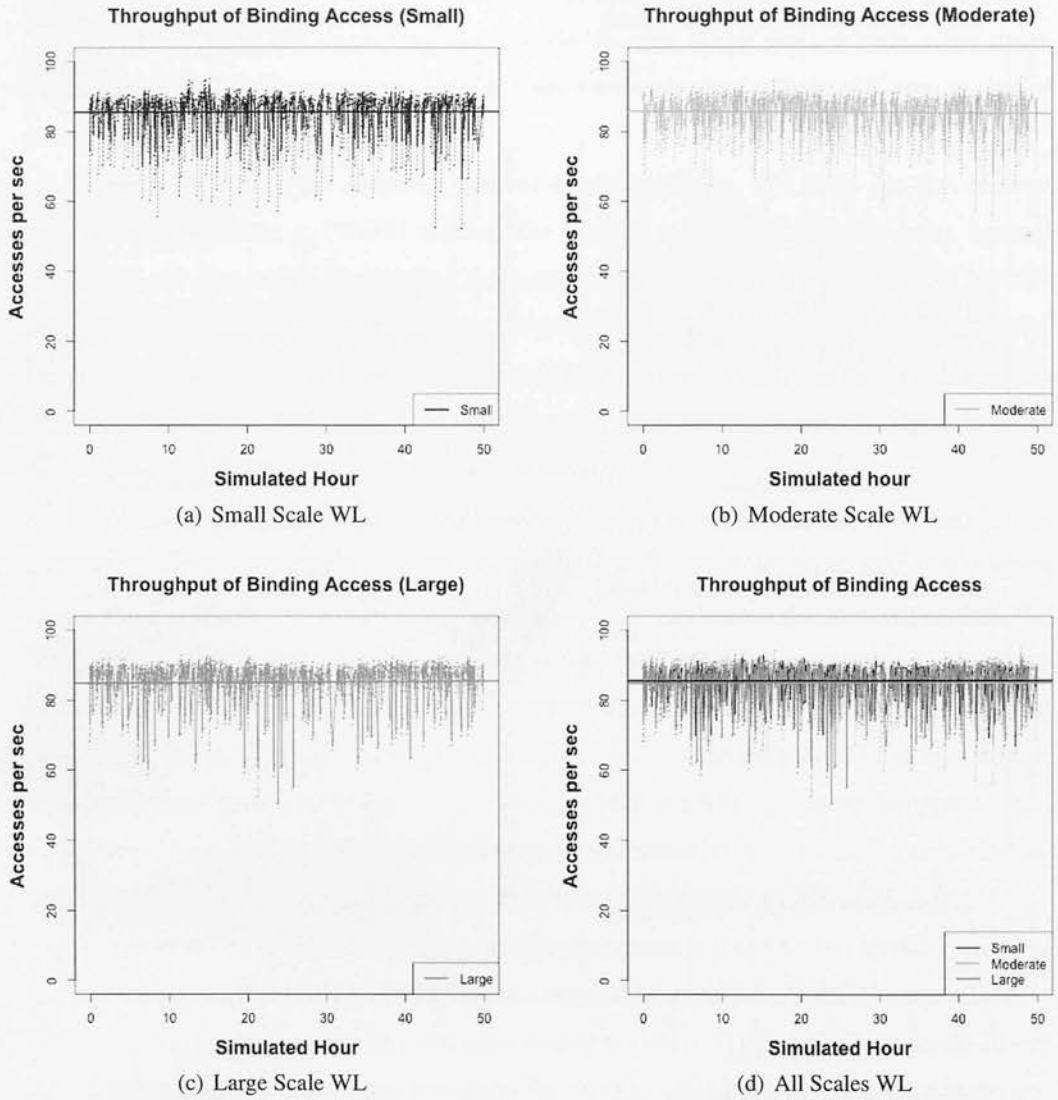


(c) Large Scale WL



(d) All Scales WL

Figure 6.19: Response time of binding access, associated with regression lines. The measures are taken under three different scales of binding access workload. In each graph, the x-axis represents the simulated time in hours. The y-axis represents the service response time in (WCT) seconds. Each data point is the mean of 10 trials, and error bars show the Confidence Intervals (*CIs*), approximately *Mean* ± *2* × *Standard Error*, which gives a range of values with 95% confident contains the true mean.



(a) Small Scale WL

(b) Moderate Scale WL

(c) Large Scale WL

(d) All Scales WL

Figure 6.20: Throughputs of binding access, associated with regression lines. The measures are taken under three different scales of binding access workload. In each graph, the x-axis represents the simulated time in hours. The y-axis represents the average number of access per (WCT) second. Each data point is the mean of 10 trials, and error bars show the Confidence Intervals (*CIs*), approximately *Mean* ± *2* × *Standard Error*, which gives a range of values with 95% confident contains the true mean.

Performance of Combined Workloads

The third group of experiments investigate the performance of the binding service under combined workloads. Two simulation threads simultaneously generate binding-creation workloads and binding-access workloads. The three different types of binding-creation workloads (*Uniform*, *Growth*, and *Burst*) combined with binding-access workloads are tested, and the scale variations are small, moderate and large[19]. Each configuration is run ten times in order to collect means and 95% Confidence Intervals, and a total 90 trials are attempted.

Figure 6.21 shows the service response time. Graphs (a)(c)(e) compare the effects of the variations of the workload scales, and graphs (b)(d)(f) compare the effects of the variations of the workload types. Two sets of curves are displayed: the upper three curves are the binding-creation performance, and the lower three curves are the binding-access performance. The figures illustrate that the binding-access speeds are (in average 4 times) quicker than the binding-creation speeds. Another observation is that the performance patterns remain the same as those in handling independent workloads, indicating that the binding service is sufficient to support the simultaneous requests of creates and reads.

The service throughputs under combined workloads are plotted in Figure 6.22, and the regression lines are associated[20]. Again, graphs (a)(c)(e) compare the performance behaviours under different scales of workloads, while graphs (b)(d)(f) compare the performance behaviours under different types of workloads. The upper three curves are the binding-access throughputs and the lower three curves are the binding-creation throughputs.

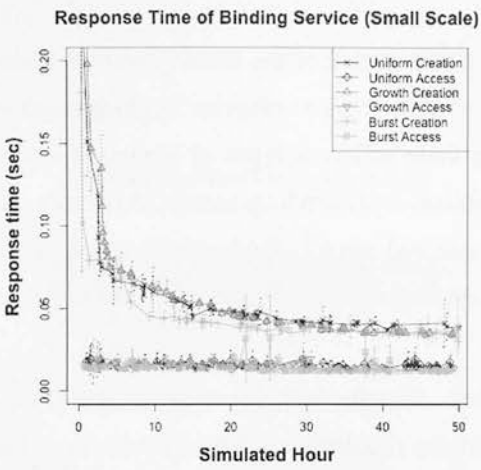
The binding-creation throughputs show the similar slow starts as before, which also slightly affect the read performance. In graphs (d) and (e), some peak performances of binding-access at the start of runs can be observed. They relate to *Large-Scale Access* workload combined *Large-Scale Growth Creation* workload. This is because when the simulation started creates and reads at the same time, the initial reads are over a small set of bindings created by that time. As a result, there is a high frequency of the required results being in memory. This infers that performance is, at least at that time, dependent on the disk traffic.

Finally, although the overall service (creation **and** access) throughputs are indeed increased, the average individual (creation **or** access) throughputs are reduced in comparison with those the isolated workloads. Notably, we did not observe any decreasing trends, and the service performance is acceptable.

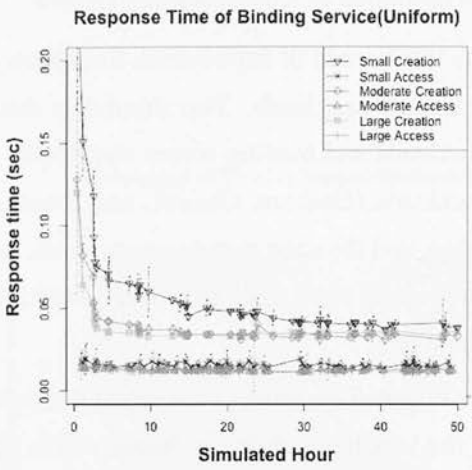
---

[19]The *Ultra Scale* workloads are omitted in this group of experiments, and we concentrate more on the normal cases

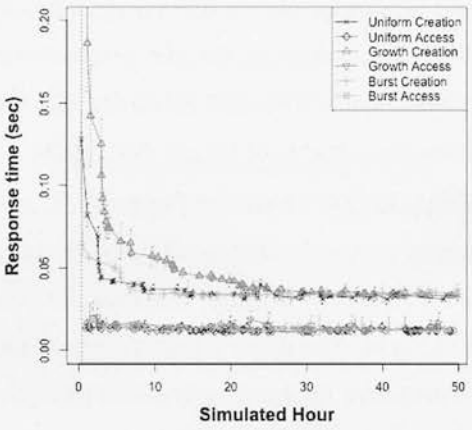
[20]Using non-linear regression model, fitted with function, $y = \frac{a \times x}{b+x} - c \times x$, by applying R nls() statement.



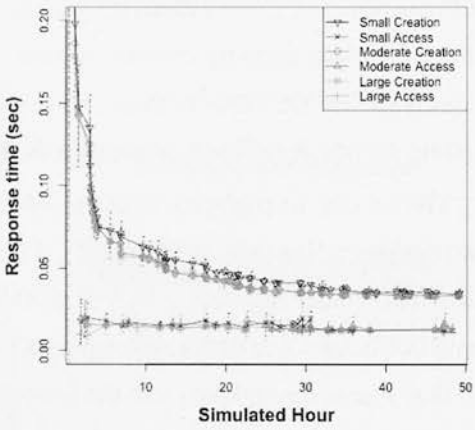
(a) Small Scale Combined WL



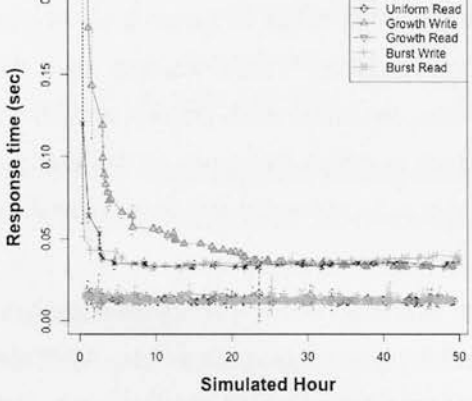
(b) Uniform Creation WL + Access WL



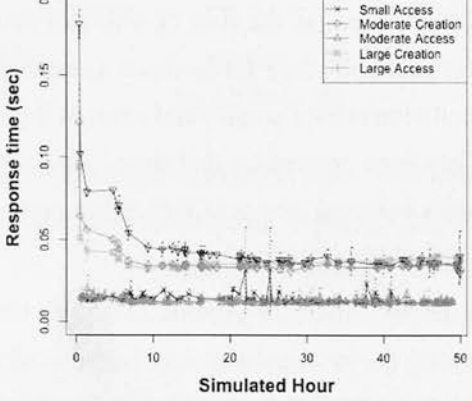
(c) Moderate Scale Combined WL



(d) Continuing Growth Creation WL + Access WL

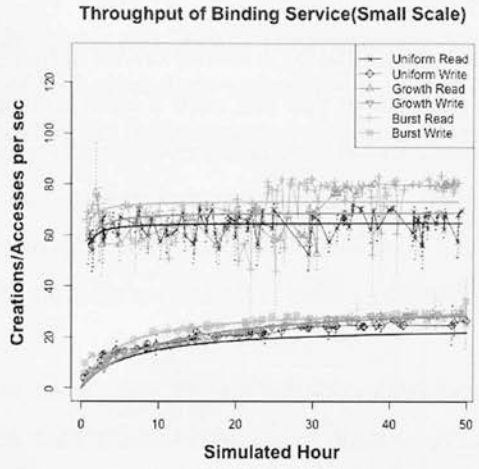


(e) Large Scale Combined WL

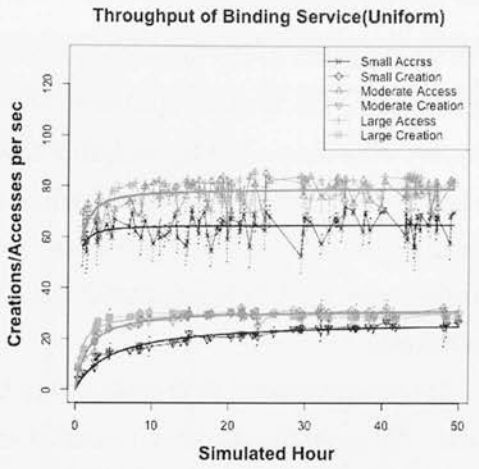


(f) Burst Creation WL + Access WL

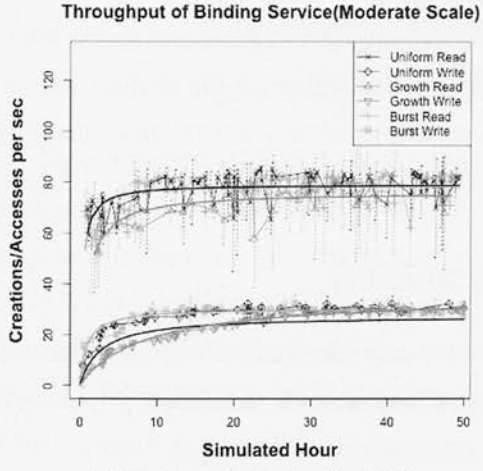
Figure 6.21: Response time of the binding service under the combined workloads. The plots are the binding Creation/Access response time in (WCT) seconds (y-axis) against the simulated hours (x-axis). The upper three curves show the response time of binding-creation. The lower three curves show the response time of binding-access. Each data point is the mean of 10 trials, and error bars show the Confidence Intervals (CIs), approximately *Mean* $\pm$ *2* $\times$ *Standard Error*, which gives a range of values with 95% confident contains the true mean.



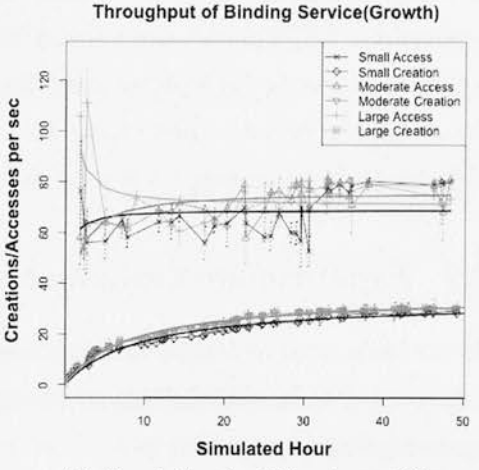
(a) Small Scale Combined WL



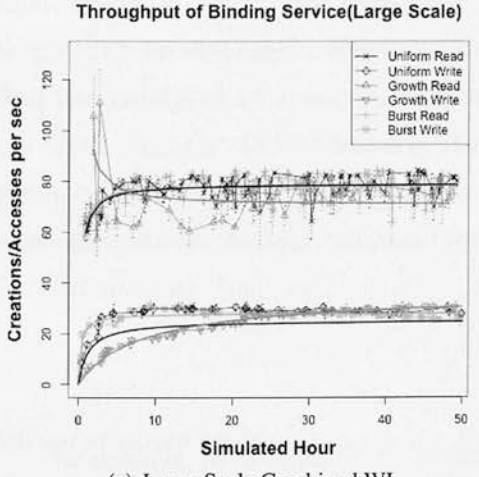
(b) Uniform Creation WL + Access WL



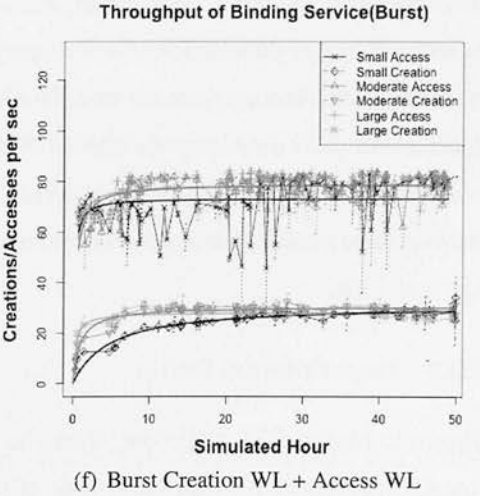
(c) Moderate Scale Combined WL



(d) Growth Creation WL + Access WL



(e) Large Scale Combined WL



(f) Burst Creation WL + Access WL

Figure 6.22: Throughputs of the binding service under the combined workloads, associated with regression lines. The plots are the binding Creation/Access per (WCT) second (y-axis) against the simulated hour (x-axis). The upper three curves show binding-access throughputs. The lower three curves show the binding-creation throughputs. Each data point is the mean of 10 trials, and error bars show the Confidence Intervals (CIs), approximately *Mean* $\pm$ *2* $\times$ *Standard Error*, which gives a range of values with 95% confident contains the true mean.

6.1.4 Conclusion

The performance of the binding service has been quantified by a model-driven simulation method. The workload patterns of updates and queries that that such a service must support were discovered by analysing existing systems. A simulation environment was established to generate the representative workloads, and binding service was tested under different types of binding-creation, access and combined workloads. The empirical results show that the binding service is robust in handling varied types and varied scale of workloads.

However, the aspects of performance that have been achieved was on a single server node. The experiments have identified that using a single server MySQL database as the binding storage was unable to handle heavy workloads. As illustrated in Chapter 2, nowadays experimental science is generating vast volumes of data, and many scientific applications, i.e., a provenance tracking service, will require to update and access bindings in high speeds and high volumes. Such applications will stress the binding system. We decided to further explore more about the scalability issues, and the next group of experiments test a strategy for scaling up the binding storage.

6.2 Experimental Evaluation of Scalability

As we introduced in Chapter 3, the Cloud is a recently emerged technology to handle large-scale data. In this study, we adopt one of the Cloud technologies, Hadoop[21], to scale up the binding storage.

Inspired by Google's Google File System (GFS) [Ghem 03] and MapReduce [Dean 04], Hadoop is designed to scale to petabytes of storage. It runs on top of the file systems built from a cluster of data nodes. On February 2008, Yahoo! Launched the world's largest Hadoop production application that runs on >10,000 cores as a Linux cluster and produces data that is used in every Yahoo! Web search query [Baez 08].

In the following experiments, we replace the single node MySQL database with a Hadoop-based binding storage, and run a series of controlled experiments to test the storage scalability.

6.2.1 Experimental Setup

Shown in Figure 6.23, a Hadoop cluster is set up, which consists of one master to run the 'master' daemons. It is the *nameñode* of the Hadoop, but it also acts as a *datanode* with regard to data storage and processing. The slave nodes only consists of the Hadoop *datanodes*, and run the 'slave' daemons. The 'master' daemons are responsible for coordination

[21] Hadoop: http://hadoop.apache.org/

and management of the 'slave' daemons, while the latter do the actual data storage and data processing work. The job tracker running on the master node controls the task tracker on each slave node to monitor the status of the parallel execution of the map/reduce tasks and report the completion of each user submitted job.

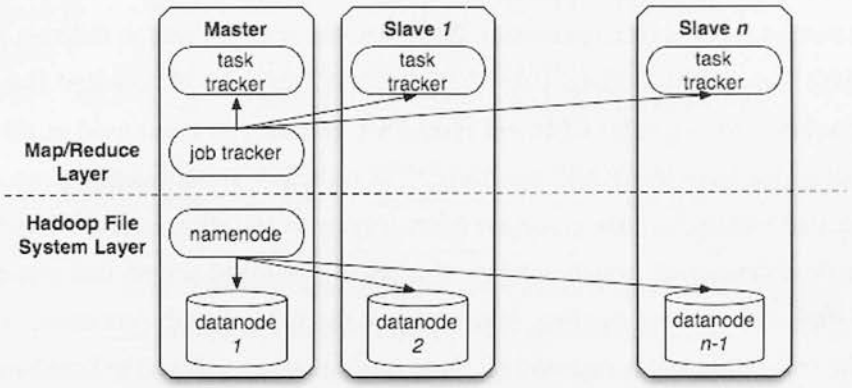


Figure 6.23: A N-node Hadoop Cluster Binding Storage

Since the workload simulator developed for earlier experiments is not capable of generating scalable workloads sufficient for testing the Hadoop-based binding storage, we implement a new workload generator based on the Hadoop Map/Reduce framework. It models only simple workload types and concentrates more on scalability evaluation. The controlled variables include the size of Hadoop clusters, the scale of binding creation, and the scale of binding space for searching. Performance in terms of elapsed time of binding-creation and binding-access are measured. In order to understand more about behaviours of Hadoop, the web service layer is excluded in this set of experiments.

On binding creation, the workload generator uses the Hadoop Map method to randomly generate simple bindings, $< id, d, m, \{a_0, a_1, \ldots\} >$, and write them into the storage. The sizes of tags range from 10 to 100 terms[22]. The terms are randomly selected from the Linux dictionary. The cluster size is N nodes, $N \in \{2,3,5,6\}$, and the total bindings generated are varied to be a total of X bytes. Each map task is assigned to create x bytes of bindings, and each Hadoop *datanode* is controlled to process z tasks, where z is the same for all *datanodes*. Thus, we have:

$$X = N \times z \times x \tag{6.5}$$

For example, to generate 10GB of bindings on a 6-node Hadoop cluster, it is necessary to assign each node 5 map tasks, and each map task is to generate 341MB $((10 \times 1024)/(6 \times 5))$ bindings.

[22] As before, we only model the binding tag behaviours, and use fix size texts for the other three binding attributes (binding *ID*, *subject*, and *object*).

On searching for bindings, the workload generator invokes the Hadoop Map method to search through the storage to find a single matched tag, and the results are written out by the Reduce method into a single file. Once again, each node is assigned the same number of parallel processes, z, and the binding search time over X bytes bindings across N nodes Hadoop cluster is measured, $N \in \{2,3,5,6\}$.

The testbed is based on a lab system. Each machine is configured as follows: Intel(R) Core(TM)2 Duo CPU 2.66GHz, RAM 2.0GB, Hard Disk 145GB, OS Red Hat 2.16.0. These machines have similar CPU and Hard Disk facilities as those used in the earlier experiments, but have less RAM capability. The machines are physically close to each other, located within the same computer laboratory, with 100Mbps network interface. A benchmark is developed, which consists of a set of command scripts that generate the binding data, search for a binding item, measure the time of both processes, and then delete the data. Each test is repeated 10 times, and the mean values, the Standard Errors (SEs) and the 95% Confidence Intervals (95% *CIs*) are calculated.

6.2.2 Experimental Results

Performance of Binding Write

Table 6.4 presents the elapsed time in seconds for generating 1GB to 10GB bindings over 4 different sizes of Hadoop clusters. The measures are taken from the time the master node initiates the Map jobs to the time the last *datanode* finishes the final Map task. Each configuration is tested 10 times to collect the means and standard errors, and a total 400 trials are attempted. These data are reproduced in Figure 6.24, which provides visualisation of the performance trends.

The left graph of Figure 6.24 plots the elapsed time in seconds (y-axis) against the total size of binding creation in gigabytes (x-axis). The graph illustrates that the binding creation time is a function of the writing load. The overall running time grows linearly as the writing load increases. It can be calculated from Table 6.4, for a 2-node cluster, the ratio for generating 10GB records to 1GB records is approximate 9.5:1. For a 6-node cluster, this ratio is 9.3:1.

The right graph of Figure 6.24 plots the binding-creation time (y-axis) against the size of Hadoop cluster (x-axis). The graph shows that the performance improves when expanding the size of clusters. For creating 1GB bindings, when adding the nodes number up to 3, the performance gain rises nearly 20 percent comparing to 2-node cluster. For creating 10GB records, when adding the nodes up to 6, the performance is about 55 percent better than 2-node cluster.

Table 6.5 shows the statistics of storage capability and throughputs of binding-creation over different sizes of clusters. The usable binding storage facility grows up to >800GB

Table 6.4: Elapsed Time of Binding Creation in Seconds

Dataset Size	2 nodes	3 nodes	5 nodes	6 nodes
1GB	79.5 (1.1)	63.6 (1.1)	46.2 (1.3)	36.2 (0.5)
2GB	151.1 (0.7)	125.7 (0.7)	97.1 (2.4)	67.0 (1.1)
3GB	222.7 (1.7)	185.7 (1.7)	122.1 (1.1)	104.7 (1.7)
4GB	294.4 (2.1)	243.1 (2.1)	155.9 (1.5)	138.7 (2.0)
5GB	377.7 (2.4)	294.6 (2.4)	233.5 (4.4)	175.7 (2.0)
6GB	450.6 (2.9)	356.0 (2.9)	276.4 (3.5)	206.9 (3.0)
7GB	521.1 (3.2)	417.3 (3.3)	317.0 (5.1)	237.9 (2.0)
8GB	603.0 (3.4)	485.3 (3.3)	380.2 (4.7)	272.3 (1.5)
9GB	673.0 (6.0)	545.2 (6.0)	415.4 (6.5)	294.5 (3.5)
10GB	757.7 (3.4)	596.7 (3.3)	463.0 (7.9)	339.7 (4.6)

Note. This table lists the elapsed time of binding-creation in seconds on 4 different sizes of Hadoop cluster. The data volumes are varied from 1GB to 10GB. Each experiment is repeated 10 times. Figures in parentheses are standard errors. The data presented here is reproduced in Figure 6.24.

Table 6.5: Throughputs of Binding Creation

	2 nodes	3 nodes	5 nodes	6 nodes
Binding Storage Capability (GB)	269.29	403.935	673.225	807.87
Number of map tasks	30	30	30	30
Tasks per node	15	10	6	5
Avg. throughputs (MB/s)	13.6	16.9	22.7	29.6

Note. These data correspond to the same set of experiments as Table 6.4, which describes the average throughput of binding creation over each experimental cluster.

in the 6-Node cluster, and a total 30 parallel processes are used to support the binding processes. The binding-creation throughput increases when more nodes are added to the cluster. The average throughput over the 6-node cluster is >2 times greater than over a 2-node Hadoop-based binding system. A total 29.6 MB/s throughput is achieved at the 6-node cluster. Consider the typical size of a binding (using the figures from Table 6.2 to calculate) is roughly 1.5KB ~4KB, which implies potentially 7,400 ~19,000 bindings per second can be processed by a 6-node cluster. This satisfies our current goal of investigation. Note that Hadoop replicates data for safety, and the replicas are generated during the initial writing. In this group of experiments, 2 replicas are used for each file. The binding-creation throughput reported in Table 6.5 do not include the Hadoop replication throughput, which means the actual throughput of the Hadoop cluster should be (2 times) higher than the binding-creation throughput.

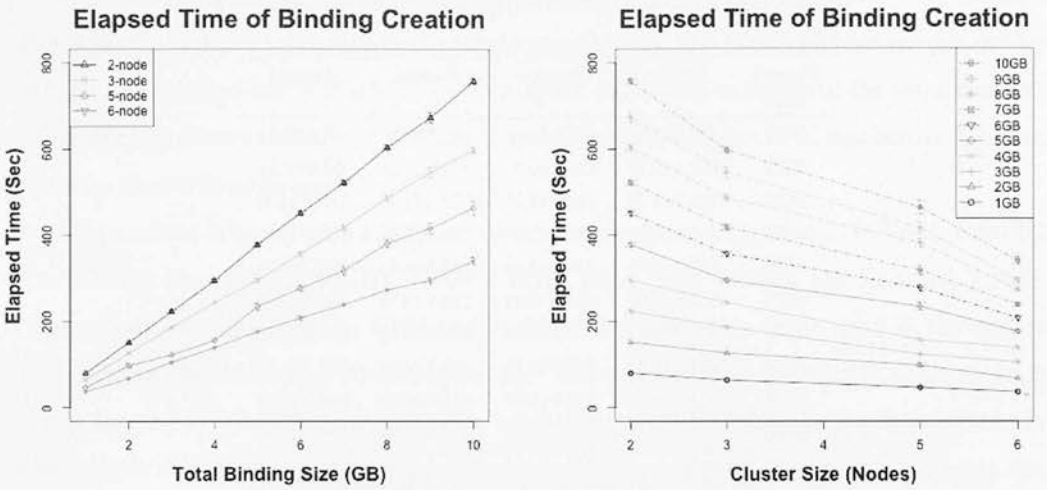


Figure 6.24: Elapsed time of binding creation in second. Graph (a) plots the elapsed time of binding-creation (y-axis) against the volume of binding created (x-axis). Graph (b) plots the elapsed time of binding-creation (y-axis) against the size of Hadoop cluster (x-axis). Each data point is the mean of 10 trials, and error bars show the Confidence Intervals (*CIs*), approximately *Mean* $\pm$ *2* $\times$ *Standard Error*, which gives a range of values with 95% confident contains the true mean.

Performance of Binding Read

Table 6.6 lists the elapsed time of binding-access in seconds for searching 1GB to 10GB binding spaces using 4 different sizes of Hadoop clusters. The measures are taken from the time the master node takes the query and initiates the Map/Reduce process to the time the final *datanode* returns results. Each configuration is tested for 10 times, the means and standard errors are calculated, and a total 400 trials are attempted. Figure 6.25 provides the graphical representations of these measures. The binding-access throughput is shown in Table 6.7.

The left graph of Figure 6.25 plots the elapsed time of binding-access in seconds (y-axis) against the volume of bindings in store (x-axis). The graph illustrates the total search time increases as the size of binding space grows. Calculated from Table 6.6, for a 2-node cluster, the search takes about 3.8 times as long at a reading load of 10GB as at a reading load of 1GB. For a 6-node cluster, the time of searching through 10GB records is 3 times as long as for searching through 1GB records.

The right graph of Figure 6.25 plots the elapsed time of binding-access in seconds (y-axis) against the size of cluster (x-axis), which shows the binding-access performance upgrades when expanding the cluster size. Using the figures in Table 6.6, it can be calculated, for searching over a 1GB bindings, the 6-node cluster is 85 percent faster than

Table 6.6: Elapsed Time of Binding Access in Seconds

Dataset Size	2 nodes	3 nodes	5 nodes	6 nodes
1GB	66.6 (2.3)	56.0 (0.7)	38.1 (0.8)	35.9 (0.8)
2GB	62.7 (2.7)	62.6 (2.9)	56.9 (1.5)	55.7 (2.7)
3GB	95.7 (2.7)	75.1 (2.7)	57.6 (1.7)	53.5 (1.3)
4GB	137.0 (1.8)	100.8 (1.2)	69.9 (2.8)	61.3 (1.4)
5GB	134.5 (2.0)	99.8 (1.6)	70.0 (1.1)	62.4 (1.9)
6GB	173.5 (2.6)	127.1 (2.4)	83.8 (1.3)	76.4 (1.6)
7GB	172.9 (1.6)	131.4 (2.4)	85.6 (0.8)	78.0 (1.4)
8GB	219.0 (2.3)	161.2 (2.4)	101.5 (1.3)	89.5 (1.0)
9GB	216.9 (2.3)	159.7 (2.0)	104.4 (1.0)	100.5 (1.0)
10GB	251.5 (2.5)	183.3 (1.7)	141.4 (1.5)	104.9 (0.8)

Note. This table lists the binding-access time in seconds over four different sizes of Hadoop clusters. The sizes of binding space are varied from 1 GB to 10 GB. Each data point is the mean of 10 trials. The figures in parentheses are standard errors. These data correspond to the Figure 6.25.

Table 6.7: Throughputs of Binding Access

	2 nodes	3 nodes	5 nodes	6 nodes
Binding Storage Capability (GB)	269.29	403.935	673.225	807.87
Number of map tasks	30	30	30	30
Tasks per node	15	10	6	5
Avg. throughputs (MB/s)	34.6	45.1	64.6	72.5

Note. These data correspond to the same set of experiments as Table 6.6, which describe the average throughput of binding read over each experimental cluster.

the 2-node cluster. For searching over 10GB records, the 6-node cluster is nearly 2 times faster than the 2-node cluster.

Table 6.7 illustrates that the average binding-access throughput is a function of the cluster size. The average binding-read throughputs are better than the average binding-creation throughput. The 6-node cluster produced 72.5MB/s binding-read throughput, which is 2.4 times of binding-write throughput. The reasons might due to (a) disk write is slower than disk read, and (b) Hadoop replicates the data when writing.

6.2.3 Conclusion

These experiments explored the potential advantages of Cloud technology and examined the scalability of Hadoop-based binding system. By using a simple configuration, (connecting machines in computer lab with normal network support,) we have achieved over 800GB binding storage, 29.6MB/s binding-creation throughput, and 72.5MB/s binding-

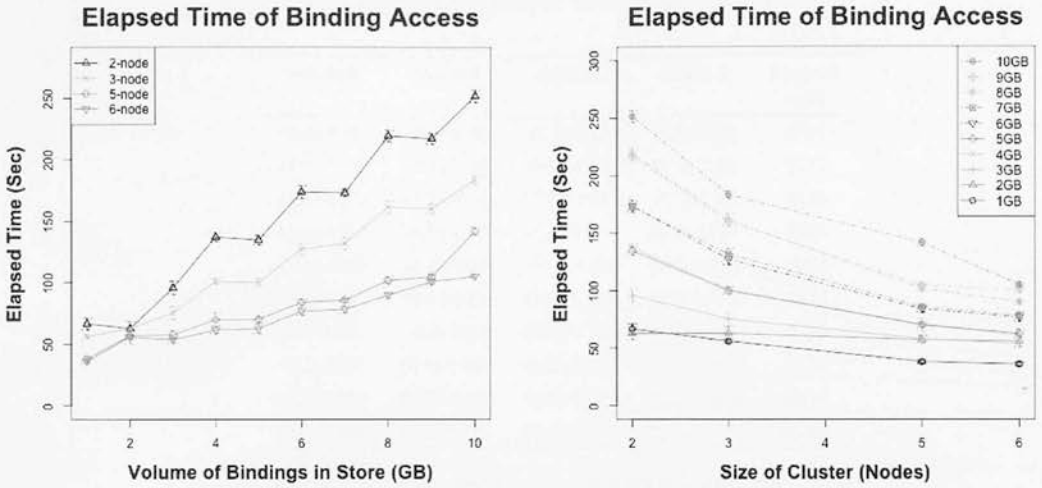


Figure 6.25: Elapsed time of binding access in seconds. Graph (a) plots the elapsed time of binding-access (y-axis) against the total volume of bindings in store (x-axis). Graph (b) plots the elapsed time of binding-access (y-axis) against the size of Hadoop cluster (x-axis). Each data point is the mean of 10 trials, and error bars show the Confidence Intervals (*CIs*), approximately *Mean* $\pm$ *2* $\times$ *Standard Error*, which gives a range of values with 95% confident contains the true mean.

access throughput at a 6-node cluster. The binding data structure is demonstrated to be suitable for parallelisation and can be handled by the Map/Reduce program. These are satisfactory for current investigation goals.

6.3 Summary

The performance of the binding service have been evaluated by empirical measurement. A number of existing scientific systems were observed and analysed to characterise the workloads that a binding service would need to support. These were then fitted to parametric statistical description of the workloads so that synthetic workloads with comparable properties could be simulated. The simulated workloads were applied to the prototyped binding service implementation to measure performance. The performance results showed that the binding service was robust to different types workloads in small to large scales.

The performance can be improved in many ways, for example, to provide composite and bulk operations so as to reduce data movements and messages between client and server. However, performance is not the only criteria for our investigation and we do not chose to support those operations to gain performance as they would introduce complexity for users and decrease take up. The performance that has been achieved is adequate for many existing scientific applications.

We have chosen to investigate the scalability problem, since the experiments have revealed that using a single server MySQL database as binding storage was not scalable, and we anticipated that special or future applications would require intensive binding processes. In the situation that scalability turns up to be a problem, we have prepared the solution to scale up the binding storage. Cloud technologies has been applied, and the performance of the Hadoop-based binding storage has been quantified. The results showed the a 6-node Hadoop-based binding storage with basic configuration was sufficient to support up to 10 gigabytes binding creation and access, which satisfied our current investigation goals.

In the next chapter, we provide an application example to show how the binding service can be used to support scientific applications.

Chapter 7

Binding Use Cases

This chapter evaluates the value of the binding approach in supporting scientific applications. Numerous information inconsistency problems caused by absence of explicit binding management were noted in Chapter 2, we now explore how such problems can be resolved by using the binding service. A binding service is installed to provide binding facilities for the EurExpress system. The binding constraint language is applied to detect inconsistency between EurExpress data and metadata. The performance is evaluated experimentally. We further use the seven scenarios introduced in Chapter 2 to examine binding usage in support of various types of scientific applications.

The chapter is organised as follows: section 7.1 presents the experimental evaluation of the binding service in supporting consistency checking of EurExpress data; section 7.2 discusses other binding applications, and finally, section 7.3 summarises this chapter.

7.1 Experimental Evaluation of Consistency Checking

7.1.1 Introduction

The overall goal of this group of experiments is to test the hypothesis that the binding approach is useful in facilitating information consistency checking, at affordable computational costs.

The use case of the EurExpress system is considered further. The following experiments test the feasibility of using binding validation operation to detect certain types of conflicts and duplications in both EurExpress data and metadata sets. The experiments measure and compare the performance efficiency of the validation operation in:

i Executing binding access;
ii Checking the binding data;
iii Checking the binding metadata; and
iv Checking both binding data and metadata.

The measures are collected under 30 different configurations by tuning 3 control variables, including 1) binding space size, 2) error patterns, and 3) error rates. Tests on each configuration are repeated 10 times, a total 300 trials are attempted, and the mean values, the Standard Errors (*SEs*) and the 95% Confidence Intervals (95% *CIs*) are calculated.

The empirical results show that the binding service extends the existing system capabilities to address the information inconsistency issues and some cases of conflicting or duplicated data and metadata can be found. The results also show that the time to validate binding information is $t_v \leq n \times a + \varepsilon$ where a is the time to access a binding and n is the number of bindings in the store, irrespective of the numbers, patterns and types of errors. In the following each experimental component is presented in detail.

7.1.2 Experimental Setup

As shown in Figure 7.1, the experimental environment extends the previous tests by establishing a connection between the binding service and a test copy of EurExpress Repository.

The experiment uses a test copy of the repository rather than the real data repository because small to large numbers of simulated errors are to be introduced into the dataset. In all other respects, the Testing EurExpress Repository is configured in exactly the same way as the EurExpress Repository. It consists of a copy of the TDB database which stores gene expression image metadata, and a file system which stores the biomedical images.

As before, a web server is set up to run the binding service. The implemented validation operation can search through the binding space and use the binding get data and get metadata operation, GetResource(), to interact with the Testing EurExpress Repository.

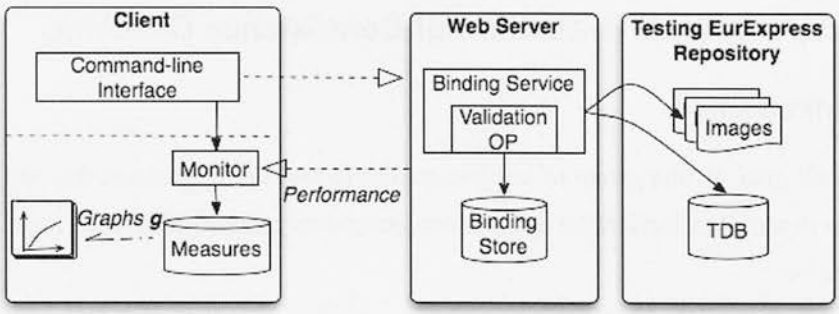


Figure 7.1: Experimental Setup

A client is installed with the command-line interface to remotely invoke binding operations. A performance monitor is installed to collect the performance measures and store them in a local database. Again, the measures are delivered to the statistical analysis tool, R, to output result graphs.

The hardware and software experimental configuration was as follows:

- Web server: Intel(R) Core(TM)2 Quad CPU 2.66 GHz, RAM 8GB, Hard Disk 1.8TB;
- Client: Intel(R) Core(TM)2 Duo CPU 3.00GHz, RAM 8GB, Hard Disk 130GB;
- The network interface: ~100Mbps;
- OS: Linux Ubuntu version 10.04.1;
- The web server is installed with Tomcat 5.5, the OGSA-DAI 4.0 server setting, and MySQL 6.0;
- The client is configured with SSJ (Stochastic Simulation in Java) 2.4, OGSA-DAI 4.0 client library, R 2.12.0, and MySql 6.0.

The experimental workflow (shown in Figure 7.2) is controlled and automated executed by scripts that:

1. Clean up the storage;
2. Set up experimental configuration variables;
3. Generate bindings;
4. Generate errors in the data and metadata sets;
5. Start the binding validation;
6. Collect and store the measure results; and
7. Finally output the results as statistical graphs.

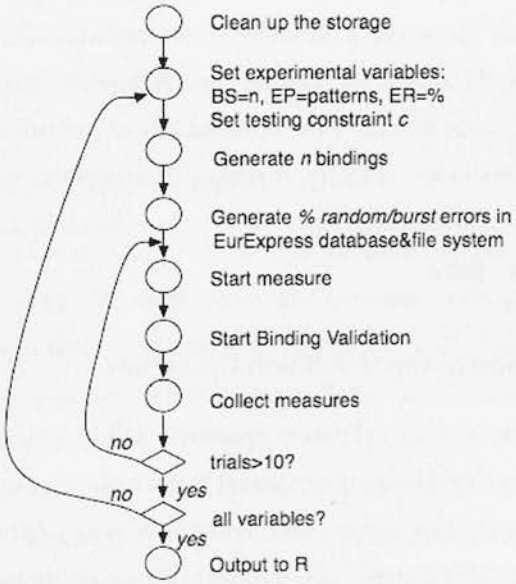


Figure 7.2: Experiment Execution Workflow

7.1.3 Experimental Data

The experimental data are the real data from the EurExpress system. Representative bindings, b_n, are created to associate the biomedical image files and their metadata records in the TDB database. The contents of b_n are shown below:

Binding Id: *UUIDs*
Binding Subject: *http://www.eurexperss.org/images/assay$_i$/image$_j$.png*
Binding Object: *http://www.eurexpress.org/query?SELECT annotation_assay$_i$_image$_j$ FROM Image_metadata_table*
Tags: *CreationDateTimeStamp, CreationUser, LastModificationDateTimeStamp, LasModificationUser, and some simple descriptions.*

The binding consistency constraints are as follows:

C_a: $\forall b$, *binding_creation_date* $\leq$ '01 Jan. 2006'
C_b: $\forall b$, *binding_creation_date* $\leq$ '01 Jan. 2006', $(b.d \neq nil) \rightarrow ((ref_d(b) \neq nil) \wedge (Unique(ref_d(b))))$
C_c: $\forall b$, *binding_creation_date* $\leq$ '01 Jan. 2006', $(b.m \neq nil) \rightarrow ((ref_m(b) \neq nil) \wedge (Unique(ref_m(b))))$
C_d: $\forall b$, *binding_creation_date* $\leq$ '01 Jan. 2006', $((b.d \neq nil) \wedge (b.m \neq nil)) \rightarrow ((ref_d(b) \neq nil) \wedge (Unique(ref_d(b))) \wedge (ref_m(b) \neq nil) \wedge (Unique(ref_m(b))))$

Intuitively, C_a simply selects a subset of bindings; C_b, C_c, and C_d claim that for each binding in the selected binding subset, the binding data or/and binding metadata must uniquely exist. In other words, C_b, C_c, and C_d are aimed to detect two types of errors: (1) non-existence of data or/and metadata, and (2) duplications of them.

The design of the four constraints serves three purposes. Firstly, it evaluates the capability of the validation operation in checking (two types of) realistic errors identified in the EurExpress system. Secondly, it measures the efficiency of executing different types of validation checking, so as to enable the observations of performance characteristics of individual validation processes. Finally, it makes it possible to compare the validation behaviour with known binding behaviours, such as pure binding access, so as to analyse the relationship between them.

7.1.4 Implementation of the Validation Operation

Recall in Chapter 5, the binding validation operation API is defined as VerifyBindings(c: String), which parses an input binding constraint statement c into executable expressions, and then verifies the expressions against the binding sets. A simple interpreter is developed and exposed via the VerifyBindings() API. Sufficient functionality is implemented to parse and evaluate the four test constraints[1].

[1] A full implementation is beyond the goal of the current work.

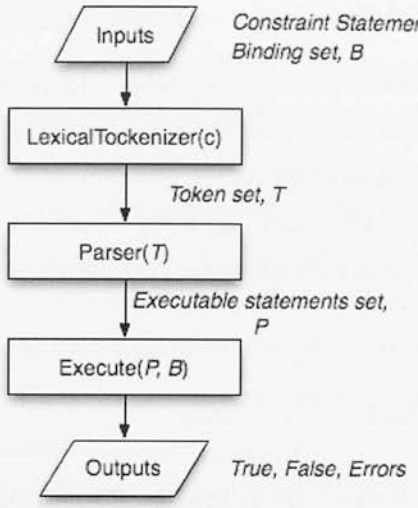


Figure 7.3: Flowchart of the Interpreter Interface of Binding Validation Operation

The program flowchart is shown in Figure 7.3. The interpreter consists of three key functions, LexicalTokenizer(), Parse() and Execute().

⇒ LexicalTokenizer (c: String) → T: String[]

For a given constraint statement c, LexicalTokenizer() takes c as an input stream, splits c into tokens and classifies the tokens into two types: *keywords* or *variables*. It returns a token set T which is a list of token-token type pairs.

⇒ Parse (T: String[]) → Exp: String[]

The result token set T is used as the input of Parse() which groups tokens into statements based on the syntax of the binding constraint language given in Table 4.1. Then, Parse() translates each statement into a (Java) executable expression and stores in a collection Exp.

⇒ Execute (Exp: String[]) → rtn: Boolean

Finally, related bindings are selected from the binding store (by using SQL), and Execute() applies the binding values to the expressions and executes them in the program context. The result returns *true* if the expression is evaluated as valid, *false* if the expression is evaluated as invalid, and *undecidable* other-wise. Error information can be obtained via a method defined by Execute class.

In a future implementation, it is anticipated that a standard language recognition tool, such as ANTLR[2], will be used. ANTLR provides a framework for constructing recognisers, interpreters, compilers, and translators from grammatical descriptions into a target language.

[2] ANTLR: www.antlr.org

7.1.5 Experimental Configurations

The experiments use three control variables (see Table 7.1). (1) The types of error patterns are the random-occurring error pattern, and the burst-occurring error pattern; (2) the size of data sets are (relevant) small, moderate and large scales; and (3) the percentage of errors in the data set are 0%, 1.76%, 10%, 25% and 50%. Notably, 1.76% is the percentage of errors detected in the EurExpress data during the preliminary study.

Table 7.1: Experimental Configurations

Error Patterns	(1) Random
(*EP*)	(2) Burst
Binding Space Size	(1) Small scale (25,079 bindings)
(*BS*)	(2) Moderate scale (136,450 bindings)
	(3) Large scale (>240,000 bindings)
Error Rates	(1) 0%
(*ER*)	(2) 1.76%
	(3) 10%
	(4) 25%
	(5) 50%

The experiments are organised as follows:

(1) For a given scale of data size and the percentage of errors introduced into the data sets, the type of error pattern is varied, and the service elapsed time of executing the binding validation operations to evaluate each of the four constraints is measured.

(2) For a given type of error pattern and the percentage of errors introduced into the data sets, the scale of data size is varied, and the service elapsed time of executing the binding validation operation to evaluate each of the four constraints is measured.

(3) For a given type of error pattern and the scale of data size, the percentage of errors introduced into the data sets is varied, and the service elapsed time of executing the binding validation operation to evaluate each of the four constraints is then measured.

7.1.6 Experimental Results

As the numbers and the types of errors in the data or metadata set are introduced by the generator, the correctness of validation can be accurately computed from the tracking information. The experiments results show us that the validation operation has successfully detected all error information introduced.

In this group of experiments, only two types of errors are examined. More types of binding errors can be verified if the consistency requirements are clearly specified by the binding constraint language, and a full implementation of the validation operation is available.

The performance of binding validation is illustrated in Figure 7.4 and 7.5. Figure 7.4 presents the plots of the elapsed time of validation operation against binding space size. The left three figures are the measures of random error patterns, and the right three figures are the measures of burst error patterns. Only error rate 1.76%, 25% and 50% are presented here, the performance characteristics of the other two configurations, error rate 0.0% and 10%, are very similar.

Figure 7.4 shows that the elapsed time of the validation operation linearly grows as the volumes of bindings increases. In all measures, the elapsed times of performing binding data and validating binding metadata are very close to the elapsed times of validating pure binding access. The elapsed times of validating both binding data and metadata are higher; yet still present the same linear trends. The regression lines suggest the time to validate binding information is $t_v \leq n \times a + \varepsilon$, where a is the time to access a binding and n is the number of bindings in the store.

Table 7.2 gives the fitted slopes and intercepts of each regression line. All residuals[3] are (randomly distributed) between [-0.08, +0.04], and have no significant variations, which suggests the fittings are acceptable.

Table 7.2: Fitting Analysis

(a) Slope ($a \times 10^{-6}$) (EP=Random)

Error Rate	Access	Validate Metadata	Validate Data	Validate Both
0.0%	9.382	9.170	9.318	9.285
1.76%	9.166	9.065	9.199	9.085
10%	9.017	9.121	8.914	9.068
25%	9.210	9.120	9.020	9.071
50%	9.019	8.917	9.160	8.864

(b) Slope ($a \times 10^{-6}$) (EP=Burst)

Error Rate	Access	Validate Metadata	Validate Data	Validate Both
0.0%	9.383	9.447	9.466	9.27
1.76%	9.557	9.399	9.442	9.393
10%	9.417	9.474	9.603	9.422
25%	9.515	9.424	9.346	9.525
50%	9.407	9.523	9.291	9.144

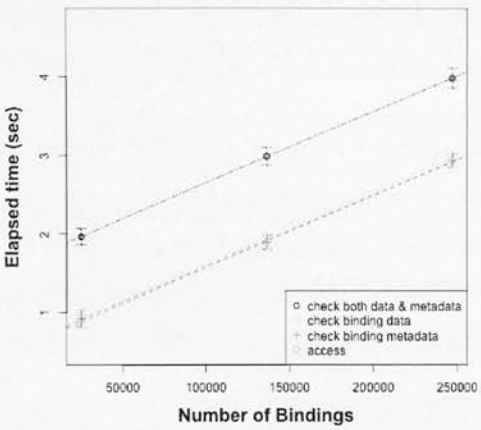
(c) y-intercept (EP=Random)

Error Rate	Access	Validate Metadata	Validate Data	Validate Both
0.0%	0.6365	0.6973	0.6448	1.742
1.76%	0.6487	0.6788	0.6915	1.738
10%	0.6566	0.6723	0.7393	1.7448
25%	0.6420	0.6668	0.7132	1.759
50%	0.6664	0.6980	0.6903	1.756

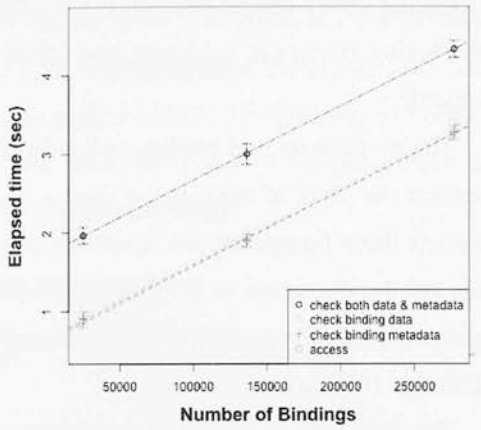
(d) y-intercept (EP=Burst)

Error Rate	Access	Validate Metadata	Validate Data	Validate Both
0.0%	0.6260	0.6557	0.6746	1.76
1.76%	0.6129	0.6865	0.6509	1.725
10%	0.6311	0.6466	0.6672	1.728
25%	0.6161	0.6576	0.6870	1.703
50%	0.6375	0.6387	0.7012	1.811

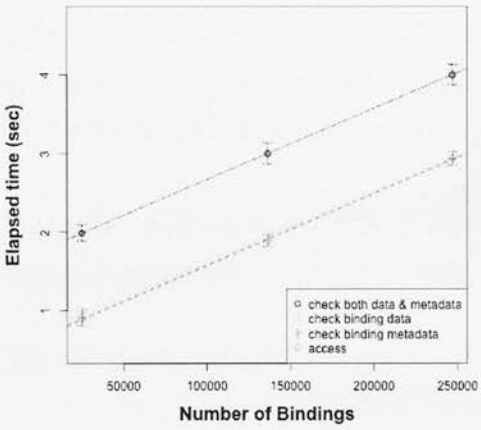
[3] Use R residuals() statement.



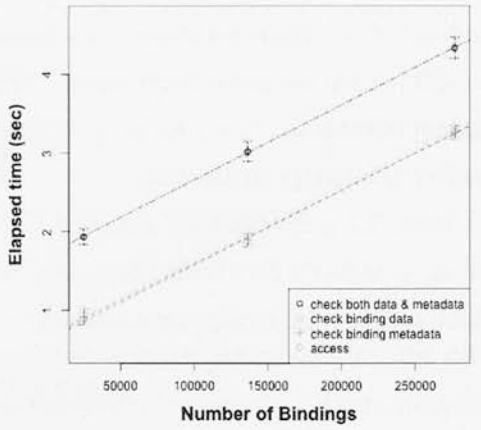
(a) **ER=1.76%, EP=Random**



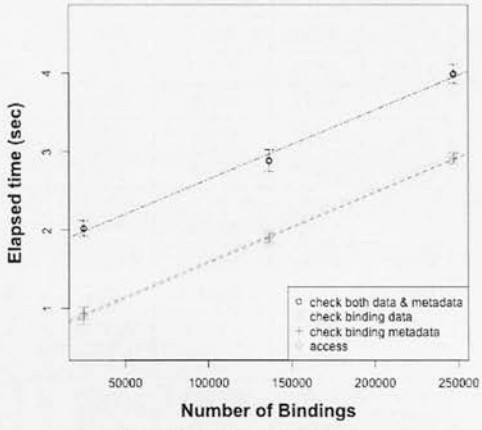
(b) **ER=1.76%, EP=Burst**



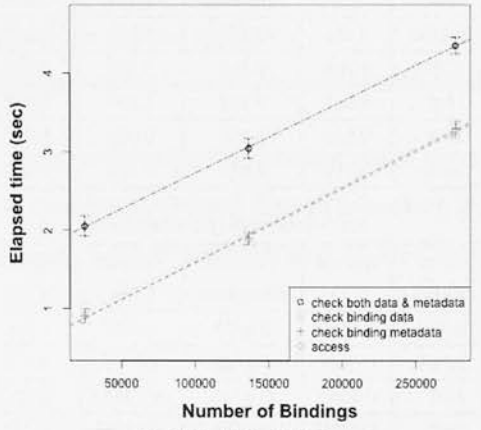
(c) **ER=25%, EP=Random**



(d) **ER=25%, EP=Burst**

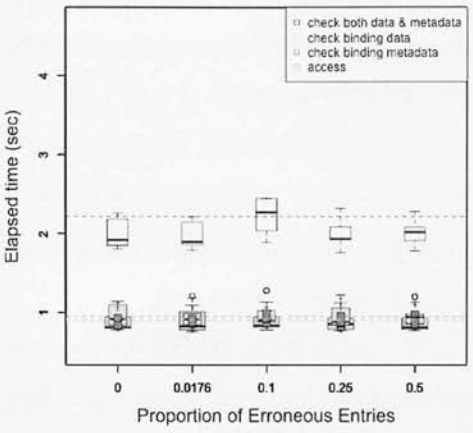


(e) **ER=50%, EP=Random**

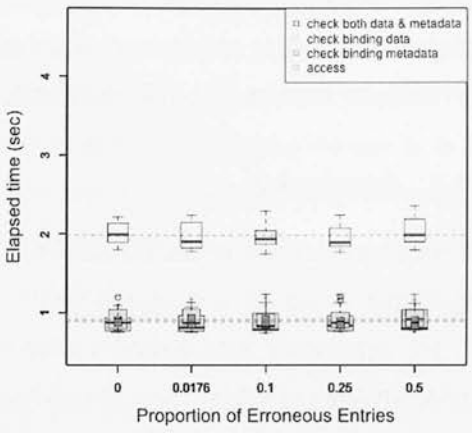


(f) **ER=50%, EP=Burst**

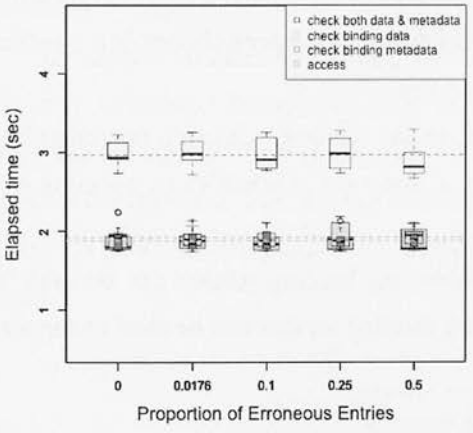
Figure 7.4: Performance of binding validation, associated with regression lines. The plots are the elapsed time (y-axis) against binding space size (x-axis). Each data point is the means of ten trials, and error bars show the Confidence Intervals (*CIs*), approximately Mean ± 2 × Standard Error, which givens a range of values with 95% confident contains the true mean.



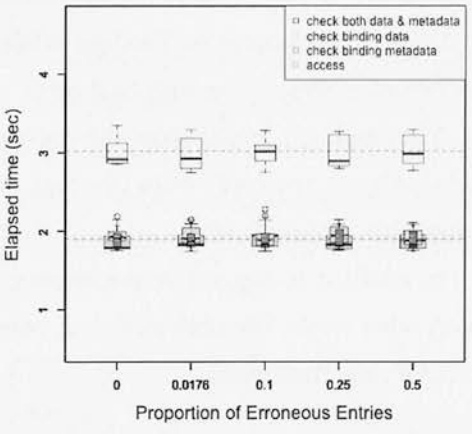
(a) **BS=25,079, EP=Random**



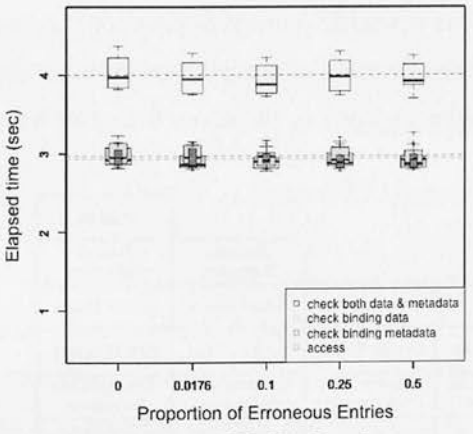
(b) **BS=25,079, EP=Burst**



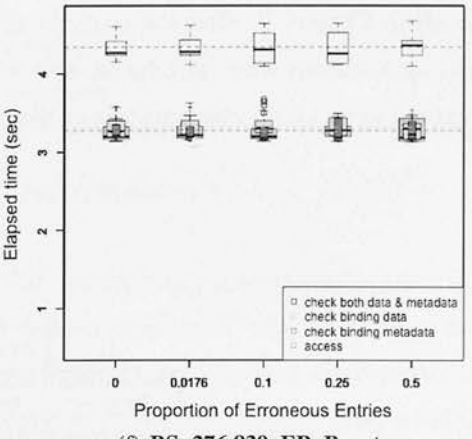
(c) **BS=136,450, EP=Random**



(d) **BS=136,450, EP=Burst**



(e) **BS=276,839, EP=Random**



(f) **BS=276,839, EP=Burst**

Figure 7.5: Performance of binding validation, associated with regression lines. The plots are the elapsed time (y-axis) against error rates (x-axis). Each boxplot shows five-number summaries of ten trials: the minimum value, lower quartile (Q1), median (Q2), upper quartile (Q3), and maximum value.

To examine the impact of error rates on the validation performance, Figure 7.5 gives the box and whisker plots of the elapsed time against the error rates for different error patterns and binding space sizes. The results shows that elapsed time of the validation operation are not sensitive to the numbers, patterns and types of errors.

7.1.7 Conclusion

Above experiments have demonstrated that the binding service can extend existing system capabilities to provide a validation facility at affordable computational cost.

The experiment used a similar setup to that of the EurExpress system. A prototype implementation of the binding validation operation has been provided to detect two types of realistic binding failures. Three controlled variables included, 1) the number of bindings in store, 2) the error pattern, and 3) the error rate, which provided 30 different system configurations. Performance of four validation behaviours have been measured under the 30 different configurations. The four validation behaviours have been chosen in a way that performance comparison with each other was possible.

The empirical results show the elapsed time of the binding validation operation linearly grows as the volumes of bindings increases, and are not sensitive to the numbers, patterns and types of errors.

In addition to support of consistency checking, the binding service can be used in many other ways. The next section discusses how binding service can be used to support scientists and their work.

7.2 Support for Binding Scenarios

Recall in Chapter 2, after the analysis of various scientific domain applications, seven binding scenarios were introduced, which were the sketches of complicated systems and captured key system characteristics. Illustrated by Figure 7.6, the seven scenarios have

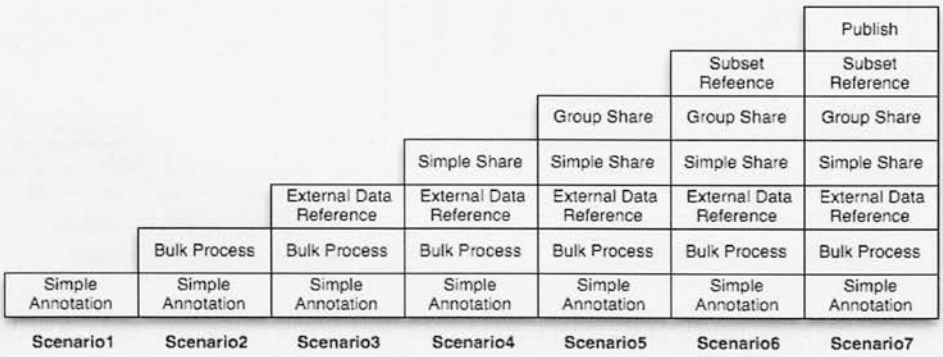


Figure 7.6: Requirements for Binding Scenarios

been built in a way that one additional key requirement was introduced at each step. Together, these scenarios represent a range of binding requirements from simple to complex cases. We note that the collection of data often begins in a project that only anticipates Scenario 1 and 2. However, if the project is successful, it, and successor projects, will progress towards the more complex scenarios. The aim of the binding service is to facilitate projects climbing this staircase as the need arises. In the rest of this section, we discuss how the binding service can be used to support these use cases.

7.2.1 Support for Binding Scenario 1

In the first binding scenario, a researcher creates files and associates tags with each that might be used later for looking up files for some task. This is the typical way for today's individual scientists to manage their data. It requires easy and efficient annotation facilities.

To support simple annotations, the binding operation CreateBinding(*ref_sub*: URI, *ref_obj*: URI, *tags*: String[]) (together with other manipulation operations) can be invoked to create (or update) bindings having file's name as *subject*, (maybe NULL *object*,) and tags for descriptions. The binding discovery operations (GetBindingsBy Subject(*ref_sub*: URI) and GetBindingsByTags(*tags*: String[])) can be used for query (by file's name and by tags).

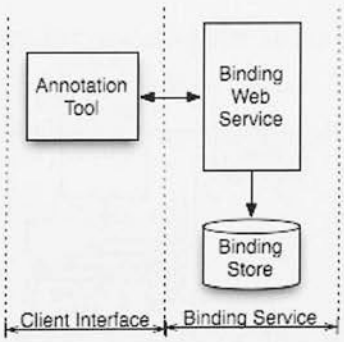


Figure 7.7: Support for Binding Scenario 1

As shown in Figure 7.7, we envisage tools that use the binding service in conjunction with other services, hiding its detailed operation from most users. For example, a tool supporting gene expression image annotators might present an interface showing the 3D gene expression images and user-selected existing available annotations from multiple sources. The annotator could add tags to classify or comment on existing annotations, or make new metadata annotations relating them to features in the image. The tool would then store these annotations in a metadata store of its choosing, and store bindings to both the metadata and the images with tags in the binding system. That binding service could

then be queried on future occasions by the tool according to each annotator's preferences. Annotators are able to draw around a region on an image and associated their annotations. The tool can capture the region, e.g. as a set of polygons or a set of run lengths, and record that in the metadata store or as tag values.

In summary, this subsection has discussed how Scenario 1 could be supported. The binding service provides sufficient facilities for simple annotations and the solution is straightforward. The next scenario extends Scenario 1 to introduce the requirement of bulk processes.

7.2.2 Support for Binding Scenario 2

In Binding Scenario 2, the researcher has collected a large number of directories of files with their metadata poorly encoded in their names. Now he wants to use the binding store to reorganise access independent from directory trees and to access relevant subsets.

To manage a large number of existing user files, a bulk information editor could be integrated as the client interface, e.g. a drag-and-drop box, as shown in Figure 7.8, such a tool might allow users to 'drag' the directories of files and 'drop' them in a location or container managed by the binding service. The software should allow users to examine the files and their metadata and make modifications, for example, to create detailed metadata, to add tags, to delete entries. It then invokes binding operations to run through the files and store the bindings.

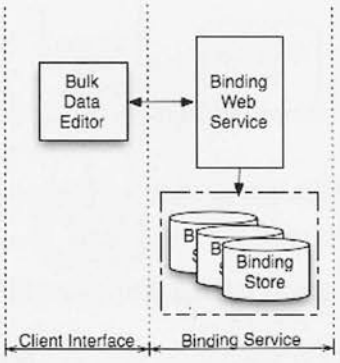


Figure 7.8: Support for Binding Scenario 2

On the binding service side, the five binding manipulation operations defined in Chapter 5 (see section 5.1.4) could be used to create and store bindings. However it would be more efficient to have *bulk* operations. This is because the bulk operations perform operations on multiple data items at once, which can often reduce the network traffic, and data sources can sometimes optimise execution of a batch of updates.

Corresponding to the five binding manipulation operations, an extension of the binding service could provide the following bulk operations:

CreateBindingsInBulk(*ref_subs*: URI[], *ref_objs* : URI[], *tags* : String[][]);
DeleteBindingsInBulk(*binding_ids*: UUID[]);
AddBindingObjectsInBulk(*binding_ids*: UUID[], *ref_objs*: URI[]) ;
InsertTagsInBulk(*binding_ids*: UUID[], *tags*: String[][]);
RemoveTagsInBulk(*binding_ids*: UUID[], *tags*: String[][]);

Each bulk operation uses Array types to model the input-lists of binding attributes. For database storage systems, each bulk operation can implement a bulk-execution SQL interface which can execute a batch of SQL statements, e.g. using storage procedures having arrays of parameters. For example, arrays of parameters can be used with the following `INSERT` statement to insert multiple bindings into the BindingTable while executing only a single SQL statement:

```
1 INSERT INTO BindingTable ( BT_bindingID, BT_subject, BT_object ) VALUES ( ?, ?, ? );
```

When bindings become too large to be processed by a single database, the binding service could use the Hadoop-based storage clusters, in which case, the binding bulk operations can have map/reduce interfaces to partition bindings onto the cluster nodes, or update them.

To conclude, this subsection has discussed how to use the binding service to support Scenario 2, in particular, how to support bulk processes. Extensional binding operations have been provided, which would allow (large number of) existing data to have binding supports easily.

7.2.3 Support for Binding Scenario 3

In Binding Scenario 3, the researcher wants to use files of their own as in Scenario 1 or Scenario 2 plus files in a reference repository and select subsets from the combination. The key requirement introduced by this scenario is to support external information references.

As shown in Figure 7.9, the binding service can be configured to communicate with remote storage systems or web services. The binding operation to support this function is the *get*-operation, GetResource(*ref*: URI), which retrieves data items from an indicated remote source. Another *get*-operation, GetResourceStatus(*ref*: URI), allows checking the availability of an indicated resource. Implemented via the data access and integration software, OGSA-DAI [Anto 07], the two *get*-operations are able to access different types of storage systems, including relational databases, XML databases, file systems, or another web data service.

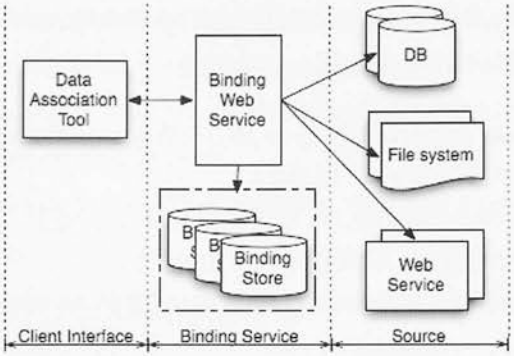


Figure 7.9: Support for Binding Scenario 3

On the client side, a data association tool can be integrated, for example, it might present an interface, on the one side showing gene expression images, on the other side showing a list of internal and external data sources, and their contents can be viewed. The tool would allow users to map images and resources to each other, and add annotations. Then, bindings could be generated and stored.

In summary, this subsection has provided the solution to Binding Scenario 3. The available binding operations for external data references and possible client interface have been discussed. Next, we will look at how to use the binding service to support information sharing.

7.2.4 Support for Binding Scenario 4

In Binding Scenario 4, the researcher wants to share some (or all) of his data with his collaborators. To support simple sharing, the binding service needs to provide the necessary access control.

The design of binding service has assumed that security would be dealt with by calls to other services (see subsection 5.1.1). Here, we are not going to discuss the security details. In Figure 7.10, we assume the client interface implements a basic user login function which allows registered users to access the system. With user information, the client tool could then retrieve bindings by user's name and bindings shared by other, for example, those tagged by 'ShareableBinding'. The tool would allow a user to view his own bindings and all shared bindings, and also allow him to mark up own bindings for sharing. The bindings marked for sharing can be associated with a system tag, e.g. 'ShareableBinding', and stored in the binding system.

Recall in Chapter 5 (see subsection 5.1.3), when defining binding data structure, we specified each binding to have a system tag 'CreationUser' to indicate the owner of the binding. When the client tool needs to retrieve bindings by user's name, the binding discovery operation, GetBindingsByTags(*tags* : String[]), can be invoked to obtain bind-

ings having matched user-name. Similarly, GetBindingsByTags(*tags* : String[]) can obtain bindings having tag 'ShareableBinding'. When stores the bindings, the operation, CreateBindings(*ref_sub*: URI, *ref_obj*: URI, *tags* : String[]), can automatically attach the tag 'ShareableBinding' to those bindings select for sharing.

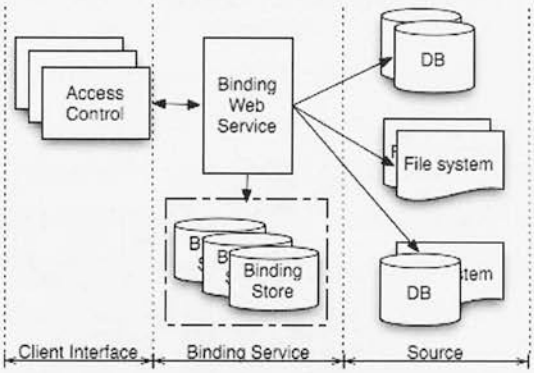


Figure 7.10: Support for Binding Scenario 4

In summary, this subsection has provided one solution for information sharing. The possible client interface has been described and the ways to map the client behaviours to the binding operations has been discussed. However, Scenario 4 has described basic binding sharing requirements, in the following, Scenario 5 will introduce more complex requirements for binding sharing.

7.2.5 Support for Binding Scenario 5

In Binding Scenario 5, a group of collaborators wants to share descriptions including referenced data.

To support sharing of binding information and group collaborations, a binding service can make use of Web 2.0-style social web sites technologies. Many e-Science practices such as myExperiment [Rour 10], and BioCatalogue [Bhag 10] have given good examples and useful experiences of using Web 2.0 to support knowledge sharing among members of communities. The public repository, myExperiment[4], is an external application of Taverna which has established a collection of scientific workflows, spanning multiple disciplines and multiple workflow systems. The myExperiment plugin for Taverna allows its workbench to search metadata of workflows and download workflows from myExperiment [Rour 10]. The BioCatalogue (as introduced in Chapter 3) uses Web 2.0, in particular, the tagging technologies to ease the service annotation and discovery.

In the case of binding applications, a Web 2.0-style interface, e.g. built by the Ruby on Rails framework[5], can be developed as a front-end to the binding service, which allows

[4] myExperiment: http://myexperiment.org

[5] Ruby on Rails: http://rubyonrails.org/

a group of collaborators to post their bindings, and view submitted bindings from other members. New post bindings and release notes can be syndicated on RSS feeds or posted on Twitter[6]. The tool would provide search facilities, e.g. using tag suggestion and tag cloud for tagging based search or browser. Retrieved bindings can be ranked by various criteria, such as based on binding popularity, similarity, or time of creation.

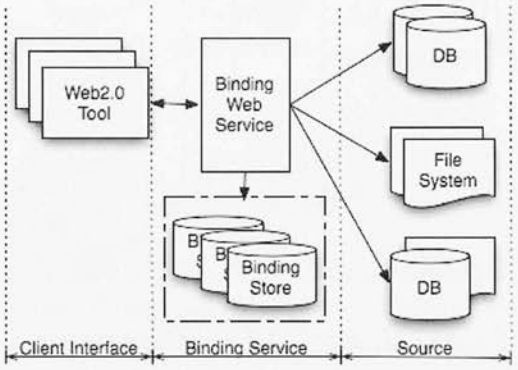


Figure 7.11: Support for Binding Scenario 5

The design of the binding service naturally supports Web 2.0 developments. Using tagging technology, the binding service is able to assign various (system) tags to bindings to indicate the status of bindings. For example, assuming that some Web2.0 standard for user identification/authentication is adopted. Then FOAF[7] statements could be used to define a group. Sharing could be permitted to an explicit access list minus an explicit exclusion list, where entries of the list are individuals or groups. These lists would be associated with tags, e.g. 'ShareWith' and 'NotShareWith', respectively. Each new submitted binding can be assigned a tag 'NewPostBinding', and each binding can be assigned a key-value pair tag 'BindingFrequency = *num*' to keep the statistics of binding access, e.g. invoking the binding operation, UpdateTags(*binding_id*: UUID, *old_tags* : String[], *new_tags* : String[]), to update the value each time when the binding being accessed. In different occasions, the client tool can invoke operation GetBindingsByTags(*tags*: String[]) to obtain desired binding sets.

To conclude this subsection, the binding solution to the group collaborations has been explored, which suggested a Web 2.0 styled interface. The design of binding service naturally supports the Web 2.0 development, and the integration methods have been discussed. In Scenario 6, we will investigate another challenging requirement.

[6]Twitter: http://twitter.com/

[7]FOAF: http://xmlns.com/foaf/spec/

7.2.6 Support for Binding Scenario 6

Binding Scenario 6 extends previous scenarios to require the referenced data to be subsets of the data in a database or parts of one or more files.

Recall the binding data structure defined in Chapter 5 (see section 5.1), a binding uses URIs to reference data and metadata associated. The syntax of a URI consists of *Query* part which enables the representations of data objects in different level of granularities. For example, it can include SQL SELECT statement to refer a subset of database tables. The URI syntax also consists of *Fragment* part which allows reference a place in a file. By using URIs, a user can associate subsets of a data resource or metadata resource. The implementation of the binding operation GetResource(*ref*: URI) supports HTTP, FTP and GridFTP network protocols, and can be used to obtain datasets.

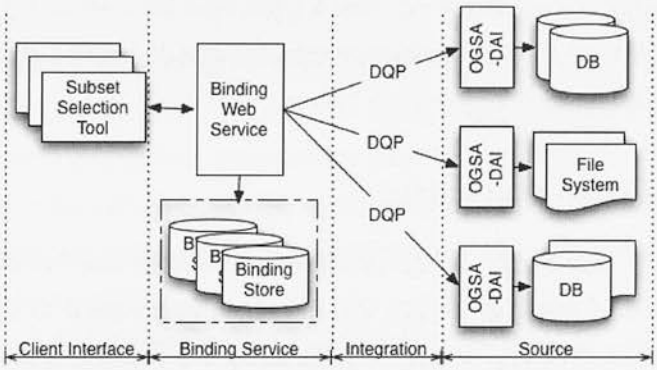


Figure 7.12: Support for Binding Scenario 6

A complex case would be that users wanted to refer to a subset of a retrieval result from multiple resources. This can be achieved by combining third-party's software which support distributed query and data integration, such as OGSA-DAI. Chapter 5 (see section 5.2) has introduced OGSA-DAI, which was a framework allows groups of activities to be chained and executed. OGSA-DAI provides a distributed query processor (DQP), which allows a single query to cite tables in multiple databases [Dobr 10]. DQP automatically parses the query and outputs a query plan which specifies the workflows to execute to get the required data from each database by using streaming technology [Dobr 10]. For example, DQP supports the following SQL statement (written in DQP grammar):

```
1 SELECT annotations FROM
2 (( SELECT annotations FROM eurexpress_db.annotation_table e WHERE e.gene = 'NRG1') UNION ALL
3  ( SELECT annotations FROM dgemap_db.annotation_table d WHERE d.gene='NRG1' ))
```

Here the annotations for gene 'NRG1' are retrieved from tables of two databases (located at separate biology labs).

The implementation of the binding service is native to OGSA-DAI, and can be extended to include the DQP facility to support distributed queries. Illustrated by Figure 7.12, the binding operation GetResource(*ref*: URI) can be extended to implement a DQP interface to resolve a URI consisting of DQP SQL statement.

A client tool can present users an interface showing available bindings with binding data and metadata as clickable links. On click those links, the client tool would invoke GetResource(*ref*: URI) to obtain the contents of the resource referenced, and displays them. The client tool would provide facilities allow user to view the contents of available resources, construct URIs and write DQP SQL statements, and create bindings using those URIs and then store them.

In summary, this subsection has discussed the binding facilities for support of subsets references to multiple data sources. A complex use case has been considered, the solution of using OGSA-DAI DQP to reference subsets from distributed resources has been explored, and the integration interface has been described.

7.2.7 Support for Binding Scenario 7

Finally, Binding Scenario 7 describes the situation when the collaborators have built a valuable collection of bindings of data and metadata, and wanted to make that public for other researchers. The requirement introduced by this scenario is to support binding publications.

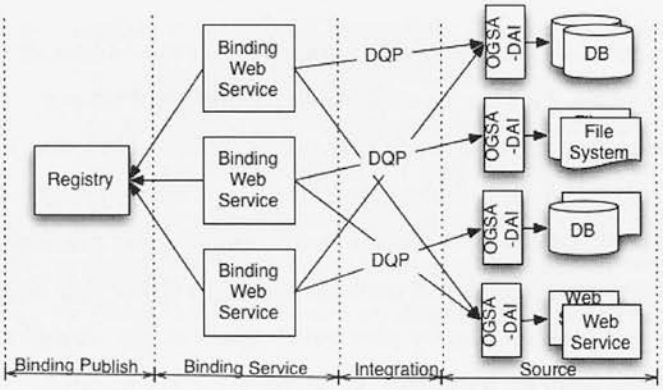


Figure 7.13: Support for Binding Scenario 7

As our purpose is to explore binding usage, we will not discuss many complex issues of data publication. We present a simple approach below and show how previous solutions can be reused and combined to resolve complicated use cases.

Figure 7.13 depicts the solution which uses a binding registry to register bindings published by individual binding services so as to support binding discovery. Importantly,

the binding registry is also a binding service, which regards the other binding services as data resources. On registering bindings, a client tool could present an interface that allows users to provide the reference of submitted binding and add tags for descriptions. The tool then could invoke operation CreateBinding() to generate a binding having subject referring to the published binding, and with tags attached. This essentially becomes the annotation problem, and the solution is similar to that in Scenario 1. The client tool can also provide a bulk binding editor for registering multiple bindings. The solution would be similar to that in Scenario 2.

However the requirement complexity introduced by a binding registry may go beyond the simple annotation and bulk processes, and is probably not be simply resolved by combining existing software. For example, it would involve following requirements on binding quality, security, and scalability:

(a) Requirement for establishing binding publication and sharing policies to encourage ethical behaviour of binding providers and binding consumers;

(b) Requirement for establishing standards, e.g. to create ontologies for binding annotation, so as to help system and users understand each other;

(c) Requirement for enforcing security, so as to know who uses the system;

(d) Requirement for curating the bindings submissions, so as to maintain the quality of the collections; and

(e) Requirement for scalability, e.g. to provide performance and efficiency when system scales up (as illustrated by the experience of publishing the Sloan Digital Sky Survey [Sing 06]).

The policy making issue in (a) is beyond this discussion (see for example, The Toronto Statement [Auth 09]). We briefly discuss the technical solutions for (b)(c)(d)(e) below:

- For (b), in Chapter 5 (see subsection 5.1.3) we have discussed that binding tags could be combined with a domain ontology or controlled vocabulary. To achieve standardisation of tags, the client tool could restrict tag inputs, meanwhile provide tag suggestion function to present users a list of legal tags to use.
- For (c), we have discussed (see subsection 5.1.1) that security would be dealt by call to other services. Depending on concrete security requirements, a binding registry can integrate a security service, such as shibboleth[8], to handle user authentication and authorisation.
- For (d), the client tool could provide curators an interface which allows them to examine submitted bindings and mark up the qualified bindings (e.g. using a tag, 'PublishableBinding') for publishing. The solution would be similar as that in Scenario 4.

[8] shibboleth: http://shibboleth.internet2.edu/

- For (e), recall scalability solution has been provided in Chapter 5. Section 5.4 has explored the Cloud technology and examined the scalability of a Hadoop-based binding storage. The result showed that it was sufficient to use Hadoop as the strategy to scale up the binding storage capabilities.

To conclude, this subsection has provided a binding registry approach to the binding publish problem. We have explored how to use a binding service to serve as a registry, and how to combine solutions of previous use cases to resolve the new problem. We have also discussed the solutions for complicated requirements for a binding registry, including issues of standardisation, security, quality, and scalability.

7.2.8 Conclusion

This section has explored the binding solutions to various binding applications. Table 7.3 provides a summary of the discussions

Table 7.3: Support for Binding Scenarios

Scenarios	Requirements	Client Interfaces	Binding Facilities	Extensions
Scenario 1	Simple Annotation	Annotation Tool	Manipulation operations Discovery operations	
Scenario 2	Scen1+ Bulk Process	Bulk Binding Editor	Discovery operations	Bulk operations
Scenario 3	Scen1,2+ External Data References	Data Association Tool	GetResource() GetResourceStatus()	
Scenario 4	Scen1,2,3+ Simple Share	User Control Access Control	Tags:CreationUser,ShareableBinding GetBindingsByTags()	
Scenario 5	Scen1,2,3,4+ Group Share	Web2.0 Tool	Tags:NewPostBinding,BindingFrequency UpdateTags(), GetBindingsByTags()	
Scenario 6	Scen1,2,3,4,5+ Subset Access	Data Viewer URI Constructor	Use URIs to reference subset of data GetResource()	OGSA-DAI DQP
Scenario 7	Scen1,2,3,4,5,6+ Publish	Registry	Tags:PublishableBinding CreateBinding()	Data Publication Policies

We have used seven use scenarios which were the sketches of existing systems and captured key binding requirements including, simple annotation, bulk processes, external data reference, information sharing, group collaboration, subset data access, and data publication. The requirement complexity has been built up at each step. Together the seven scenarios represent simple to complex application use cases. For each scenario, we reviewed the binding facilities for supports of the requirement, described possible client interface, and explained how to map the client behaviours to the binding service behaviours. For some special issues, we also discussed possible extensions for the binding service.

The analysis has shown that the binding service could be used to serve various purposes, and the solutions to the seven simple use scenarios could be combined to address more complicated requirements.

7.3 Summary

The binding requirements and challenges identified in use cases that are introduced at the beginning of the thesis have been addressed. The value of binding approach in supporting scientific applications have been demonstrated.

Firstly, an experimental analysis of a realistic binding application has been presented. The experiments have examined the capability of the binding service in providing validation facilities to the EurExpress test system. The results had shown that the binding service was able to detect realistic consistency failures embedded in scientific data at affordable computational cost.

To explore the binding usage in other types of scientific applications, the solutions of using binding service to support seven representative use scenarios were discussed. The analysis results have shown that the binding service could be flexibly extended or combined to serve a range of different types of scientific applications.

In the next chapter, we continue to evaluate the design of the binding service by comparing it with the state-of-the-art approach.

Chapter 8

Comparison with State-of-the-Art

Chapter 5 has presented the design of the binding service. This chapter evaluates the design by comparing it with related work. In Chapter 3, we have introduced the Linked Data approach. This semantic web approach addresses similar design challenges to those addressed by binding approach – how to support linkages of data from large scale distributed sources. Several design features of the two approaches are comparable, i.e., both define a data structure to represent a relationship as an association of subject with object, and both use URIs to reference external objects. This raises questions, "whether the two approaches are essentially the same?" In this chapter, a detailed comparison of two approaches is provided, which covers five perspectives including, data structure, binding expressions, storage mechanisms, query facilities, and communication protocols. Understanding this level of detail will serve many purposes, such as to show how internal capabilities impact on the external behaviours, and to guide implementations and systems integration.

The chapter is organised as follows: section 8.1 describes the comparison framework; section 8.2 compares the data structure; section 8.3 discusses how to use RDF for binding representations; section 8.4 compares the differences in the storage mechanisms; section 8.5 compares the query facilities; section 8.6 compares the communication capabilities; and finally section 8.7 summaries the chapter.

8.1 Comparison Framework

Figure 8.1 shows the comparison framework. We assume the inputs are the binding requirements, in particular, those have been captured by the binding formal model described in Chapter 4, and the outputs are computational capabilities. We compare five aspects of the Linked Data realisations of the binding requirements with the design of the binding service. We assume two aspects are equal if they give the same outputs for any identical inputs. This implies that to prove the two entities unequal it is only necessary to find out one significant controversial case.

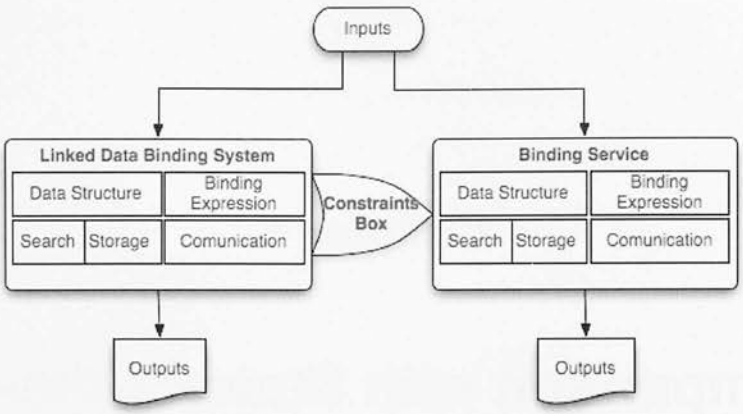


Figure 8.1: Comparison Framework. Given binding requirements as inputs, five aspects of the Linked Data approach and the binding service approach are compared. A constraint box contains constraint-rules for mapping model behaviours from the Linked Data to the binding service.

Recall the four principles of the Linked Data approach specified by Tim Berners-Lee:

1. Use URIs as names for things;
2. Use HTTP URIs so that people can look up those names;
3. When someone looks up a URI, provide useful information, using the standards RDF, SPARQL;
4. Include links to other URIs so that they can discover more things.

Intuitively, the Linked Data model seems more comprehensive than the binding service model, for example, the underlying RDF language is declared to be expressive. This, on the other hand, suggests that more complexities and conflicts might be introduced in the Linked Data model. We explore the hypothesis that the Linked Data model can be narrowed down into the binding service model by constraining its computational behaviours. To test the hypothesis, a constraint box is added, which contains constraint-rules for mapping model behaviours from the Linked Data to the binding service. The comparison aims at finding out what such constraint rules should be.

8.2 Comparison of Data Structure

The Linked Data makes use of the RDF data model. The Resource Description Framework (RDF)[1] is a W3C standard model for data interchange on the Web. RDF extends the linking structure of the Web to use URIs to name the relationships between things as well as the two ends of the link, which is usually referred to as a 'triple'. The linked structure forms a directed, labelled graph, where the edges represent the named link between two

[1] W3C RDF Primer http://www.w3.org/TR/2004/REC-rdf-primer-20040210/#rdfxml

resources, represented by the graph nodes. In the form of an RDF triple, it consists of a subject, an object and a predicate.

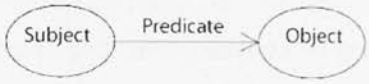


Figure 8.2: RDF Data Model

Recall the data structure of the binding service, which consists of a global unique identifier, BindingID, a reference pointing to the data (the so-called binding subject), a reference of the associated metadata (the so-called binding object), and a set of tags that are used to annotate the meaning of the binding. The data type of references is URI, of tags is string[].

While it seems the RDF subject and the object could be mapped to the binding subject and the binding object, respectively, and the RDF predicate may correspond to the binding tags, there is no property in the RDF triple corresponding to the binding identifier which is explicitly defined in the binding data model.

If the binding identifier is unnecessary, it can be removed then we have a perfect mapping. But is it removable? Let us drill down to the semantic level. The RDF data model is RDF-subject-oriented. An RDF triple gives a statement of a binary relationship from the subject pointing to the object, which is one-way direction. In contrast, the binding data model is binding-oriented. A binding defines the association relationship of the subject and the object, which is two-way direction. Defining a binding requires at least two RDF statements. For example, the RDF statements for a binding b of dataset d_1 and metadata m_1 could be:

$< Subject >$	$< Predicate >$	$< Object >$
d_1	$is\_described\_by$	m_1
m_1	$describes$	d_1

More comprehensive descriptions for binding b will create the following RDF statements:

$< Subject >$	$< Predicate >$	$< Object >$
b	$has\_a\_subject$	d_1
b	has_a_object	m_1
b	$is\_created\_by$	$Alison$
b	$has\_creation\_time$	$2010-12-11$

A binding is not an atomic RDF statement. In RDF, a binding identifier is useful to reference a binding relationship which is defined by multiple RDF statements (triples). This implies the first constraint-rule in the constraint box:

Constraint Rule 1: A binding relationship is globally identifiable.

A binding identifier is assigned to a group of inter-connected RDF nodes, and a binding pattern can be identified. Figure 8.3 shows an example. The left graph depicts an RDF graph where binding relationships have no special structures, while in the right graph a binding can be easily identified by assigning a binding identifier node. To assign identifiers to binding RDF triples is also useful for parallelisation processes when dealing with scalability. For example, it would then be easy to break the RDF binding records and partition them onto distributed nodes of a Cloud [Cast 09].

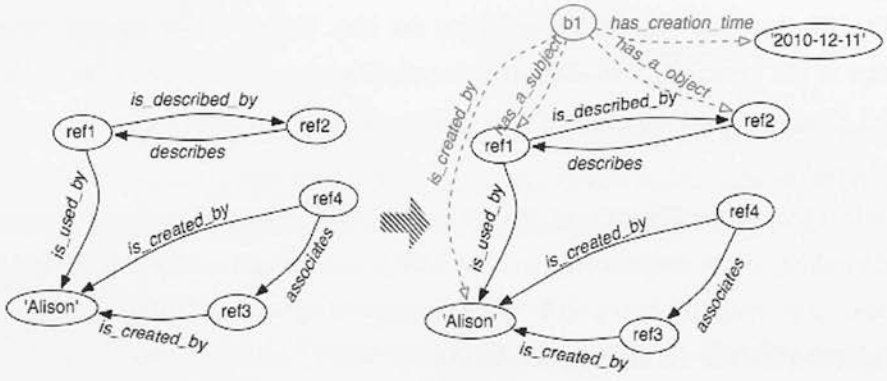


Figure 8.3: Identify binding patterns from an inter-connected RDF graph

A binding identifier differs from a rdf:ID specified by RDF/XML syntax. The rdf:ID attribute on an RDF node element can be used instead of rdf:about and gives a relative RDF URI reference. Each rdf:ID attribute value pair against in-scope base URI (xml:base) is checked to be unique within a single RDF/XML document. For example, if ref:ID="name", the value "name" can only appear once in the scope of an xml:base value or document. However, rdf:ID can not check the uniqueness beyond the scope of one RDF/XML document, neither can be an identifier of a group of RDF triples.

Discussions of various approaches to getting IDs for (a group of) RDF triples have been taking place in the Semantic Web and RDF communities, including the proposal of Named Graphs. Named Graphs is the approach that has multiple RDF graphs in a single document/repository and naming them with URIs[2]. The Named Graphs vocabulary RDFG is provided to describe graphs and relations between graphs. The technology providing serialisation for the Named Graphs in known as TriX (Triples in XML) and TriG (a plain text format for serialising Named Graphs alternative to the XML-based TriX syntax) [Carr 03, Carr 05]. The binding approach and the Named Graphs share similar views regarding identification of a group of RDF statements. However, using Named Graphs to represent bindings, further constraints would be needed to ensure that every triple in a Named Graph, that was needed to describe the binding in full was reachable and uniquely defined as being part of this binding.

[2]Named Graphs: http://www.w3.org/2004/03/trix/

8.3 Comparison of Binding Expression

Let us look at how to express bindings in RDF. A typical binding supported by a binding service can be, *b*:

Binding Id: $73b2a8ee-dd76-47eb-962c-08f540f313d$

Binding Subject: $http://www.dgemap.org/images/embryo029.png$

Binding Object: $http://www.dgemap.org/annotation/query?embryoId=029;disorder=yes$

Binding Tags: '*embryo*', '*annotation*', '*creationtime*: 2010-03-10'

An RDF graph representation of *b* is given in the top graph of Figure 8.4. Following the discussion in the previous subsection, the *Binding Id* is introduced as an individual RDF node in addition to the *Binding Subject* node and the *Binding Object* node. In the given example, the three nodes are represented by RDF URI references, while the URI of the *Binding Subject* pointing to a biological image, the URI of the *Binding Object* containing query fragment referring to a set of tuples in a relational database, and the URI of the *Binding Id* is used to locate the binding in a pre-defined XML namespace. For simplicity, the *Binding Tags* are represented as RDF plain literals, and left ungrouped without using RDF bags, sequences or lists. Finally, the binding property names, 'subject', 'object', and 'tag', are specified as XML Qualified Names (QNames, a type of CURIE). The RDF/XML syntax and the underlying triples in Notation3 (N3) formats[3] are also given in the Figure 8.4[4].

To provide useful tagging facilities, the RDF expression for representing key/value pair tags is examined. Depicted by Figure 8.5, to express a key/value pair tag, 'creationtime'='2011/05/10', an RDF blank node is introduced to link the key, 'creationtime', with the typed value, '2010/02/10'. Both 'creationtime' and '2011/05/10' are represented as RDF literal nodes.

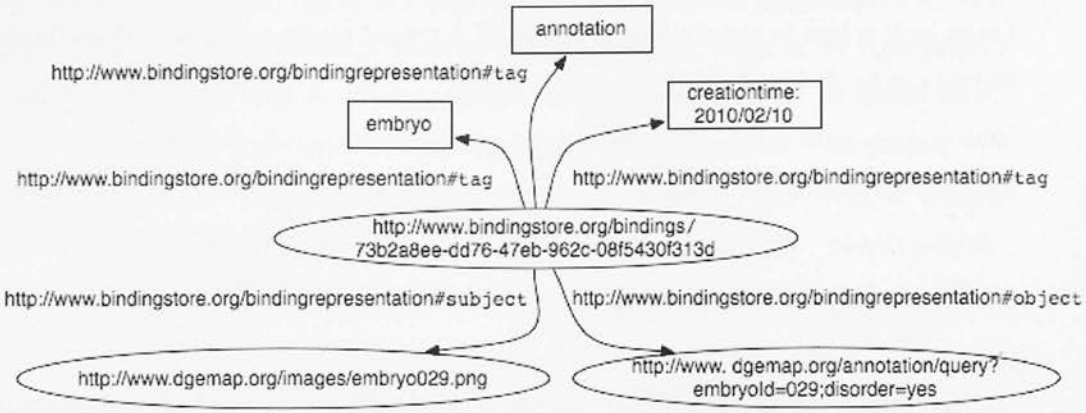
These examples show one way in which to map the binding representation onto the RDF data model without losing binding capabilities. What has been understood is that bindings can be fully expressed in RDF by introducing appropriate structures, i.e., defining the binding pattern, adding namespaces, and specifying data types. In other words, binding representation adds a structured layer on top of the RDF data model. This implies the second constraint-rule in the constraint box:

Constraint Rule 2: A binding can be viewed as a structured RDF graph.

[3] Notation3 is a shorthand non-XML serialisation of RDF, which is much more compact and readable than XML RDF notation.

[4] Validated by W3C RDF validator: http://www.w3.org/RDF/Validator/

An RDF graph for the sample binding:



The expression in RDF/XML syntax is given as follows:

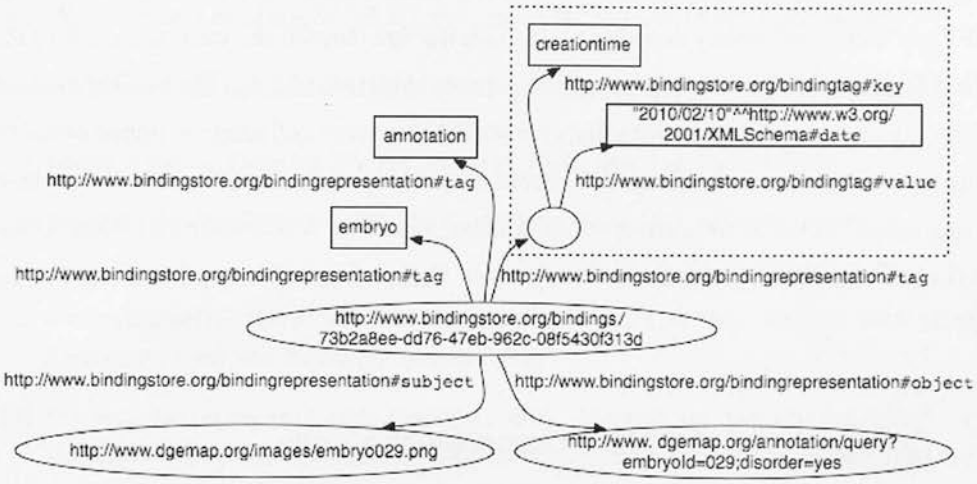
```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:binding="http://www.bindingstore.org/bindingrepresentation#"
         xmlns:bindingtag="http://www.bindingstore.org/bindingtag#">

  <binding:id rdf:about="http://www.bindingstore.org/bindings/73b2a8ee-dd76-47eb-962c-08f5430f313d">
      <binding:subject rdf:resource="http://www.dgemap.org/images/embryo029.png"/>
      <binding:object  rdf:resource="http://www.dgemap.org/annotations/query?embryoId=029;disorder=yes"/>
      <binding:tag>embryo</binding:tag>
      <binding:tag>annotation</binding:tag>
      <binding:tag>creationtime:2010/02/10</binding:tag>
  </binding:id>

</rdf:RDF>
```

The underlying RDF triples in N3 formats are below. Note each triple is broken into three lines, representing the *subject*, the *predicate* and the *object*, respectively.

```
<http://www.bindingstore.org/bindings#73b2a8ee-dd76-47eb-962c-08f5430f313d>
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
        <http://www.bindingstore.org/bindingrepresentation#id> .

<http://www.bindingstore.org/bindings#73b2a8ee-dd76-47eb-962c-08f5430f313d>
    <http://www.bindingstore.org/bindingrepresentation#subject>
        <http://www.dgemap.org/images/embryo029.png> .

<http://www.bindingstore.org/bindings#73b2a8ee-dd76-47eb-962c-08f5430f313d>
    <http://www.bindingstore.org/bindingrepresentation#object>
        <http://www.dgemap.org/annotation/query?embryoId=029;disorder=yes> .

<http://www.bindingstore.org/bindings#73b2a8ee-dd76-47eb-962c-08f5430f313d>
    <http://www.bindingstore.org/bindingrepresentation#tag>
        "embryo" .

<http://www.bindingstore.org/bindings#73b2a8ee-dd76-47eb-962c-08f5430f313d>
    <http://www.bindingstore.org/bindingrepresentation#tag>
        "annotation" .

<http://www.bindingstore.org/bindings#73b2a8ee-dd76-47eb-962c-08f5430f313d>
    <http://www.bindingstore.org/bindingrepresentation#tag>
        "creationtime:2010/02/10" .
```

Figure 8.4: An RDF graph of the sample binding, the RDF/XML expression and the underlying triples.

An RDF graph for a key/value pair binding:



It is a validated expression[4], and the RDF/XML representation is:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:binding="http://www.bindingstore.org/bindingrepresentation#"
         xmlns:bindingtag="http://www.bindingstore.org/bindingtag#">

    <binding:id rdf:about="http://www.bindingstore.org/bindings/73b2a8ee-dd76-47eb-962c-08f5430f313d">
        <binding:subject rdf:resource="http://www.dgemap.org/images/embryo029.png"/>
        <binding:object  rdf:resource="http://www.dgemap.org/annotations/query?embryoId=029;disorder=yes"/>
        <binding:tag>embryo</binding:tag>
        <binding:tag>annotation</binding:tag>
        <binding:tag rdf:nodeID="creationtime"/>
    </binding:id>

    <rdf:Description rdf:nodeID="creationtime">
        <bindingtag:key>creationtime</bindingtag:key>
        <bindingtag:value rdf:datatype="http://www.w3.org/2001/XMLSchema#date">2010-02-10</bindingtag:value>
    </rdf:Description>

</rdf:RDF>
```

Figure 8.5: The RDF graph and RDF/XML expression for the sample binding containing a key/value pair tag.

The structured binding RDF records could become useful in a variety of computations, such as validation processes (e.g. to support binding identification and validation within RDF graphs), statistical analysis (e.g. to support aggregation calculations), and distributed computing (e.g. to partition bindings onto distributed storage). For example, SPARQL is an RDF query language that performs RDF triple matching, but it does not support aggregation functions, such as average and maximum value calculation [Pahl 08]. Such statistical facilities have to be resolved at the data structure level. Solutions provided by the Semantic Web communities include the RDF Data Cube vocabulary[5] and the Vocabulary

[5]The RDF Data Cube: http://publishing-statistical-data.googlecode.com/svn/trunk/specs/src/main/html/cube.html

of Interlinked Dataset (voiD)[6]. The RDF Data Cube vocabulary is an extension of existing RDF vocabularies, including SKOS[7], SCOVO[8], vioD, FOAF[9] and Dublin Core[10]. The RDF Data Cube vocabulary supports multi-dimensional data on the web, and enables the Linked Data applications to publish various aspects of statistical data. The vioD provides a vocabulary and a set of instructions that enables the discovery and usage of linked datasets. Being given a structure, a binding RDF record is able to include above vocabularies to hold the aggregated values of the corresponding binding resource, which allows a Linked-Data-based binding service to compute these values and perform (binding-based) statistical analysis.

8.4 Comparison of Data Storage Mechanism

This section examines how information is stored by the two approaches. The design of binding service in Chapter 5 did not specify the storage mechanism. However, a relational database interface has been provided by the prototype implementation. On the other hand, the Linked Data approach in general assumes RDF store and SPARQL query methods (refer to the third Linked Data principle). In the following, we compare the computational facilities provided by the relational binding database and by a Linked Data realisation of a binding RDF store.

The binding service database schema has been given in Chapter 5 (Figure 5.6). Figure 8.6 shows sample records and illustrates how bindings are stored in the database tables.

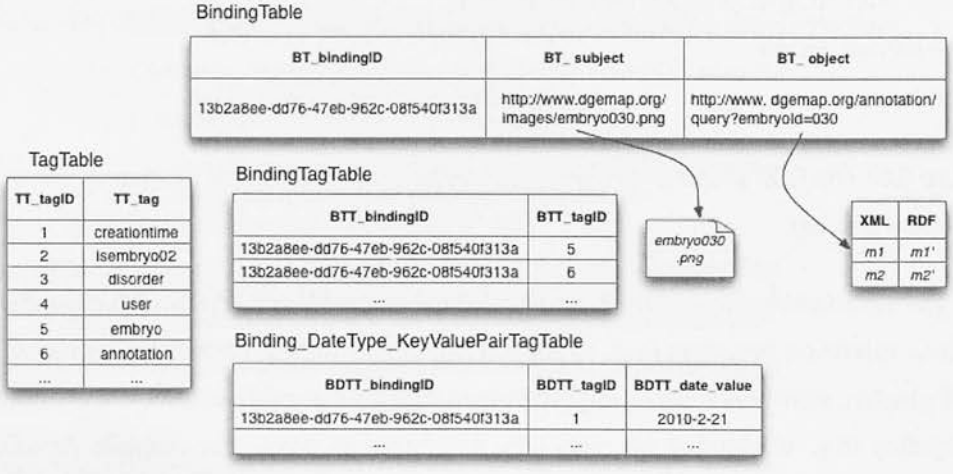


Figure 8.6: The Storage Mechanism of the Binding Service

[6]vioD: http://vocab.deri.ie/void/
[7]SKOS: http://www.w3.org/2004/02/skos/
[8]SCOVO: http://sw.joanneum.at/scovo/schema.html
[9]FOAF: http://xmlns.com/foaf/0.1/
[10]Dublin Core: http://purl.org/dc/terms/

Recall in the binding service, storing a binding is through the binding creation operation:

⇒ CreateBinding (*binding_Id*: UUID, *ref_sub*: URI, *ref_obj*: URI, *tags*: string[])

An execution of the operation triggers a series of transactions in a relational database:

[1] Insert a tuple, {*binding_Id*, *ref_sub*, *ref_obj*}, into the *BindingTable*;

[2] Tokenise the binding tags, and insert new appearing tags into the *TagTable*;

[3] For each single tag, insert a tuple, {*binding_Id*, *tag_Id*}, into the *BindingTagTable*, where the *binding_Id* and the *tag_Id* are foreign keys of the *BindingTable* and the *TagTable*, respectively;

[4] For each key/value pair tag, insert a tuple, {*binding_Id*, *tag_ID*, *tag_value*}, into the *Binding_*(**datatype**)*_KeyValuePairTagTable*, again, the *binding_Id* and the *tag_Id* are foreign keys of the corresponding tables.

The functionality of binding creation can be simulated by using the Linked Data technology. Illustrated by Figure 8.7, the bindings are kept as RDF triples in an RDF-triple store, typically a relational database. (An alternative method is to store the bindings in RDF/XML files. The RDF triple store is used as it has more comparative features.)

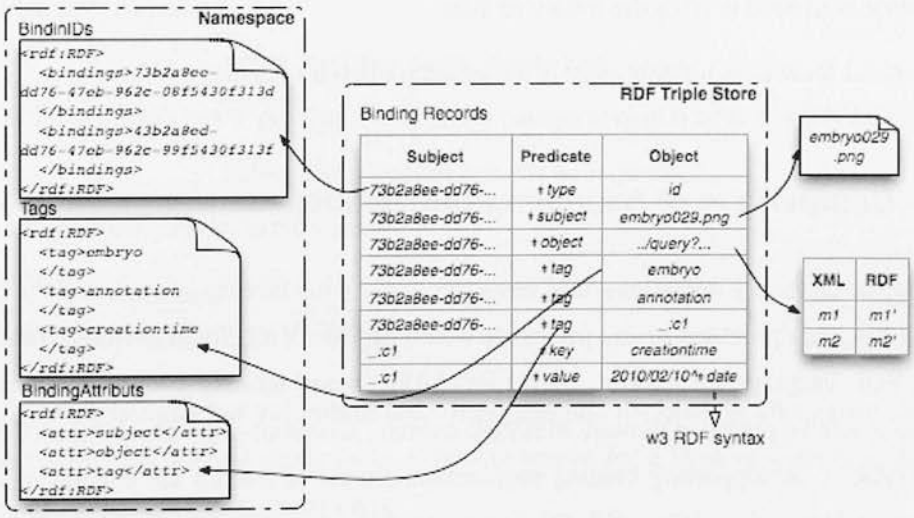


Figure 8.7: The Storage Mechanism of an RDF Binding Store

While the bindings can be managed by the RDF triple store, their dependent information – the namespaces *BindingIDs*, *Tags*, *BindingAttributes*, and RDF *schemas*, are stored separately, typically in web accessible file systems. Referencing these namespaces is through HTTP. Given this mechanism, a binding creation operation would involve the following actions:

[1] Update the namespace, *BindingIDs*, and insert the *binding_Id*;

[2] Tokenise the binding tags, update the namespace *Tags*, and insert the new appearing tags;

[3] Insert a set of triples into the binding RDF Triple Store. For the given example, the triple set are: {{*bindin_Id, rdf:type, id*}, {*id, binding:subject, ref_sub*}, {*id, binding:object, ref_obj*}, {*id, binding:tag, tag*$_1$}, ..., {*id, binding:tag, blank_node*$_1$}, {*blank_node*$_1$, *binding:key, tag*$_{n+1}$}, {*bland_node*$_1$, *binding:value, tag_value*$_{n+1}$}, ...}.

In a Linked-Data-based binding system, bindings and their dependent knowledge are loosely coupled. Contrariwise, the binding service built on top of a relational database system can reliably sustain the consistency of bindings and their dependent knowledge through transaction controls. Transaction control is one of the important facilities provided by relational database products, which maintains the persistent states of the system, ensures the software can handle unexpected failures, and supports concurrent processes. Lacking of such transaction management, a Linked-Data-based binding system is easily left to be inconsistent. For example, the updates of a binding tag may fail to be propagated to the namespace, and vice-versa. A binding store should provide consistent binding information, and need to obey the following rule:

Constraint Rule 3. A binding store is a transaction-based system.

8.5 Comparison of Searching Capabilities

A group of discovery operations have been specified by the binding service, and the realisation has been provided by the prototype which exposes a relational database interface using SQL language for queries. In a Linked-Data-based binding system, the look-up functions will be provided through SPARQL queries. To compare the capabilities of SQL and SPARQL in supporting binding requirements, a set of queries are examined. According to Gray's Law [Gray 07], 20 queries would be broad enough to cover the most important questions the researchers wanted the data system to answer. This work uses fewer queries, but characteristic queries, such as point queries, range queries, and null queries, are included.

The binding RDF records for performing SPARQL queries are given in Appendix A, which use the expressions shown in Figure 8.4 and 8.5. The same information is created in a testing binding database for performing SQL queries. A sample binding record has been given in Figure 8.6. In the following, we examine the query expressions and query results of the two languages against the same enquires.

Q1: "Give me the bindings linking to the metadata, m_1." This query is useful when obtaining all data resources or binding annotations related to a specific metadata resource. In the binding service, this is achieved by operation:

⇒ GetBindingsByObject (*ref_obj*: URI)

Suppose m_1 is `'http://www.dgemap.org/ annotations/query?embryoId=030'`, and a SQL query can be constructed as follows: [11]:

```
SELECT BT_bindingID AS bindingId
  FROM BindingTable
 WHERE BT_object = 'http://www.dgemap.org/annotations/query?embryoId=030';
```

A SPARQL statement to get bindings having object m_1 is:

```
PREFIX  binding: <http://www.bindingstore.org/bindingrepresentation#>
SELECT ?bindingId
  FROM <binding.rdf>
 WHERE {?bindingId   binding:object   <http://www.dgemap.org/annotations/query?embryoId=030> .}
```

The two expressions present similar structures: both use `SELECT` clause for data projection, `FROM` clause to indicate the query scope, and `WHERE` clause for query criterion. Differently, the SQL `WHERE` syntax is:

`WHERE` *column_name* **operator** *value* ;

while the SPARQL `WHERE` provides basic graph pattern to match against RDF data graph:

`WHERE` *subject predicate object* .

The two queries give the same results[12]:

bindingId
13b2a8ee-dd76-47eb-962c-08f5430f313a

Q2: "Get all images not yet annotated." By definition, the binding objects are allowed to be empty. The second query looks at how to search for a binding without an object. The SQL statement for the question is:

```
SELECT BT_bindingID AS bindingId
  FROM BindingTable
 WHERE BT_object IS NULL;
```

[11] We only project binding identifiers in this query for clarity.

[12] The SQL queries are processed by MySQL database v6.0. The SPARQL queries are processed by Twinkle: http://www.ldodds.com/projects/twinkle/.

The SPARQL statement follows:

```
PREFIX binding: <http://www.bindingstore.org/bindingrepresentation#>
SELECT ?bindingId
 FROM  <binding.rdf>
WHERE {?bindingId  binding:object  '' . }
```

Note, in SPARQL, an empty string can be used for searching the Null values. In SQL, Null values are normally treated differently from other values. Null is defined by ISO SQL standard[13] as different from an empty string and the numerical value 0, and special operators, `IS NULL` and `IS NOT NULL`, are provided for testing whether data is, or is not, Null.

Nevertheless, both queries give the correct answers and retrieve the bindings having no metadata associated[11,12]:

bindingId
53b2a8ee-dd76-47eb-962c-08f5430f313a

Q3: "Which bindings link to the data resource d_1." This query is useful when obtaining all bindings and metadata related to a data resource of interest. The related operation specified by the binding service is:

⇒ GetBindingsBySubject (*ref_sub*: URI)

Suppose d_1 is given as `'http://www.dgemap.org/images/embryo030.png'`. A SQL query to get the bindings associated with d_1 is:

```
SELECT BT_bindingID AS bindingId
  FROM BindingTable
 WHERE BT_subject = 'http://www.dgemap.org/images/embryo030.png';
```

An equivalent SPARQL query is:

```
PREFIX binding: <http://www.bindingstore.org/bindingrepresentation#>
SELECT ?bindingId
  FROM <binding.rdf>
 WHERE {?bindingId binding:subject <http://www.dgemap.org/images/embryo030.png> .}
```

Both queries match two bindings[11,12] referring to the same data source:

bindingId
13b2a8ee-dd76-47eb-962c-08f5430f313a
43b2a8ee-dd76-47eb-962c-08f5430f313d

More comprehensive information can be retrieved from the binding collection by querying binding tags.

[13] ISO/IEC (2003). ISO/IEC 9075-2:2003, "SQL/Foundation". ISO/IEC. Section 8.7: null predicate.

In the binding service, an API is specified:

⇒ GetBindingsByTags(*tags*: string[])

The API handles single tag query (including wildcard match), a list of tags query, key/value pair tag query, and range value query. Implementations of more sophisticated query capabilities, such as approximated query based on similarity measures, and *top-k* query based on possibility algorithms, can be exposed through this API. In the following, we examine tag related queries.

Q4: "Get all bindings related to embryo." This is a single-tag query, and the SQL query string is as follows:

```
SELECT b.BT_bindingID AS bindingId, t.TT_tag AS tag
  FROM BindingTable b, BindingTagTable bt, TagTable t
 WHERE b.BT_bindingID = bt.BTT_bindingID
    && bt.BTT_tagID   = t.TT_tagID
    && t.TT_tag LIKE '%embryo%';
```

The SPARQL query string is:

```
PREFIX binding: <http://www.bindingstore.org/bindingrepresentation#>
SELECT ?bindingId  ?tag
  FROM <binding.rdf>
 WHERE{?bindingId  binding:tag  ?tag.
FILTER regex(str(?tag), "embryo")
}
```

In SPARQL, strings are considered to match the pattern if any substring matches the pattern, which is analogous to the wildcard-match in SQL languages. In the SQL statement we use the wildcard `'%'` to achieve the same results[12]:

bindingId	tag
13b2a8ee-dd76-47eb-962c-08f5430f313a	embryo
73b2a8ee-dd76-47eb-962c-08f5430f313d	ISembryo029

Both queries give two solutions. The second binding is retrieved since the tag `'ISembryo029'` contains the substring 'embryo', which is regarded as matched string in SPARQL. In the case an exact match is required, a special symbol, `'$'`, should be attached at the end of the string to be matched[14]. For example, in the given query, `'embryo$'` shall be use in the SPARQL statement, which would give the first binding only.

Q5: "Are there any imagines related to disordered embryo, annotated by HDBR? An embryo head is sufficient." This maps to a tag-list query. Suppose a user-input tag-list: {'*disordered*', '*embryo*', '*head*', '*annotated by HDBR*"}. In the binding service,

[14] Different SPARQL parsing tool may specify this differently.

the operation GetBindingByTags() returns bindings containing *any* of tags in the tag-list, which in essence assumes logic-OR between the specified tags. A Logic-AND tag-list query can be achieved by taking the intersection of the datasets returned from two or more GetBindingByTags(). There are no restrictions that Logic-AND couldn't be implemented by GetBindingByTags(). To do so, the Logic-OR query can be similarly implemented by taking the union of the datasets obtained from two or more GetBindingByTags().

Here, we assume Logic-OR between the tags, and construct the following query statements. A Logic-AND query can be easily constructed, but is omitted here.

The SQL statement is:

```
SELECT b.BT_bindingID AS bindingId, t.TT_tag AS tag
  FROM BindingTable b, BindingTagTable bt, TagTable t
 WHERE b.BT_bindingID = bt.BTT_bindingID
    && bt.BTT_tagID   = t.TT_tagID
    && ( t.TT_tag LIKE '%disordered%' ||
         t.TT_tag LIKE '%embryo%' ||
         t.TT_tag LIKE '%head%' ||
         t.TT_tag LIKE '%annotated by HDBR%')
ORDER BY bindingId;
```

And the SPARQL string for the intended query is:

```
PREFIX binding: <http://www.bindingstore.org/bindingrepresentation#>
SELECT  ?bindingId  ?tag
  FROM  <binding.rdf>
 WHERE {?bindingId binding:tag ?tag.
FILTER (regex(str(?tag), "disordered") ||
        regex(str(?tag), "embryo") ||
        regex(str(?tag), "head") ||
        regex(str(?tag), "annotated by HDBR"))
}
ORDER BY ?bindingId
```

Note, SPARQL provides `FILTER` operation to test strings, based on regular expressions, which is analogous to the `LIKE` style tests in SQL. The `ORDER BY` (and `DISTINCT`) clause is also supported in SPARQL, as the results shown below[12], which provides the same functionality as that in SQL.

bindingId	tag
13b2a8ee-dd76-47eb-962c-08f5430f313a	annotated by HDBR
13b2a8ee-dd76-47eb-962c-08f5430f313a	embryo
13b2a8ee-dd76-47eb-962c-08f5430f313a	head
43b2a8ee-dd76-47eb-962c-08f5430f313d	disordered
73b2a8ee-dd76-47eb-962c-08f5430f313d	ISembryo029

Q6: "Give all bindings created by Alison." This is an example of the key/value-pair tag query. A SQL interpretation of this question is:

```
SELECT b.BT_bindingID AS bindingId, t.TT_tag AS key, bst.BSTT_string_value AS value
  FROM BindingTable b, Binding_StringType_KeyValuePairTagTable bst, TagTable t
 WHERE b.BT_bindingID = bst.BSTT_bindingID
    && bst.BSTT_tagID = t.TT_tagID
    && bst.BSTT_string_value = 'Alison';
```

Recall in the binding database (the relations schema in Figure 5.6), key/value pair tags are treated differently from single-word tags and stored in separate tables. In this query, Binding_StringType_KeyValuePairTagTable is included for looking up the user, '*Alison*'. The query retrieves the following binding[12]:

bindingId	key	value
13b2a8ee-dd76-47eb-962c-08f5430f313a	user	Alison

The SPARQL interpretation of the question follows:

```
PREFIX binding: <http://www.bindingstore.org/bindingrepresentation#>
PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?bindingId  ?key ?value
 FROM  <binding.rdf>
WHERE {?bindingId  binding:tag   ?tag.
       ?tag        binding:key   ?key;
                   binding:value ?value.
FILTER regex (?value,  "Alison$")
}
```

In the RDF binding representation, key/value pair tags also need special treatments. Recall in Figure 8.5, blank nodes are used to express key/value pair tags. The graph pattern to match such bindings are given as:

```
?bindingId  binding:tag   ?tag.
?tag        binding:key   ?key;
            binding:value ?value.
```

The query returns the same set of bindings yet in a slightly different form[12]:

bindingId	key	value
13b2a8ee-dd76-47eb-962c-08f5430f313a	user	Alison^^http://www.w3.org/2001/XMLSchema#string

Here, '`Alison^^http://www.w3.org/2001/XMLSchema#string`' is an RDF typed literal with the datatype '`http://www.w3.org/2001/XMLSchema#string`'.

Q7: "Get all images after 2010-02-10." This is an attempt of a range query.

The SQL statement is:

```
SELECT b.BT_bindingID AS bindingId, t.TT_tag AS key, bdt.BDTT_date_value AS value
  FROM BindingTable b, Binding_DateType_KeyValuePairTagTable bdt, TagTable t
 WHERE b.BT_bindingID = bdt.BDTT_bindingID
    && bdt.BDTT_tagID = t.TT_tagID
    && bdt.BDTT_date_value > '2010-02-10';
```

This time, Binding_DateType_KeyValuePairTagTable is included to look up date values. The query retrieves the following results[12]:

bindingId	key	value
13b2a8ee-dd76-47eb-962c-08f5430f313a	creationtime	2010-02-21
53b2a8ee-dd76-47eb-962c-08f5430f313a	creationtime	2010-02-21
73b2a8ee-dd76-47eb-962c-08f5430f313d	creationtime	2010-02-21

The equivalent SPARQL query is constructed as follows:

```
PREFIX binding: <http://www.bindingstore.org/bindingrepresentation#>
PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xsd:<http://www.w3.org/2001/XMLSchema#>
PREFIX : <http://www.bindingstore.org/tags/>
SELECT  ?bindingId  ?key  ?value
  FROM  <binding.rdf>
 WHERE {?bindingId  binding:tag   ?tag.
        ?tag        binding:key   ?key;
                    binding:value ?value.
FILTER (isBlank(?tag) &&
        regex(str(?key), "creationtime") &&
        xsd:date(?value)>"2010-02-10"^^xsd:date)
}
ORDER BY ?bindingId
```

This query selects the following datasets[12], which is the desired retrieval:

bindingId	key	value
13b2a8ee-dd76-47eb-962c-08f5430f313a	creationtime	2010-02-21^^http://www.w3.org/2001/XMLSchema#date
53b2a8ee-dd76-47eb-962c-08f5430f313a	creationtime	2010-02-21^^http://www.w3.org/2001/XMLSchema#date
73b2a8ee-dd76-47eb-962c-08f5430f313d	creationtime	2010-02-21^^http://www.w3.org/2001/XMLSchema#date

As illustrated by Figure 8.8, SPARQL applies Cartesian product of the two sets on the shared entity, when matching multiple query graph patterns. This process is analogous to the Inner-Join statement in SQL language. In this example, the first triple pattern in the WHERE clause,

```
WHERE { ?bindingId  binding:tag  ?tag. FILTER   (isBlank(?tag)) }
```

retrieves the RDF graphs that describe: "a bindings has a tag which is a blank node". The second triple pattern:

```
WHERE { ?tag        binding:key   ?key;  binding:value ?value.
FILTER   regex(str(?key), "creationtime") && xsd:date(?value)>"2010-02-10"^^xsd:date) }
```

retrieves the RDF graphs that describe: "the key of a tag is 'creationtime' and the value of that tag greater than '2010-02-10' ". Thereafter, two sets of RDF graphs are joined

to produce the final results. Apart from the Inner-Join, the semantics of SQL Left outer-Join and Right outer-Join are captured by SPARQL OPTIONAL and UNION operations, respectively[15].

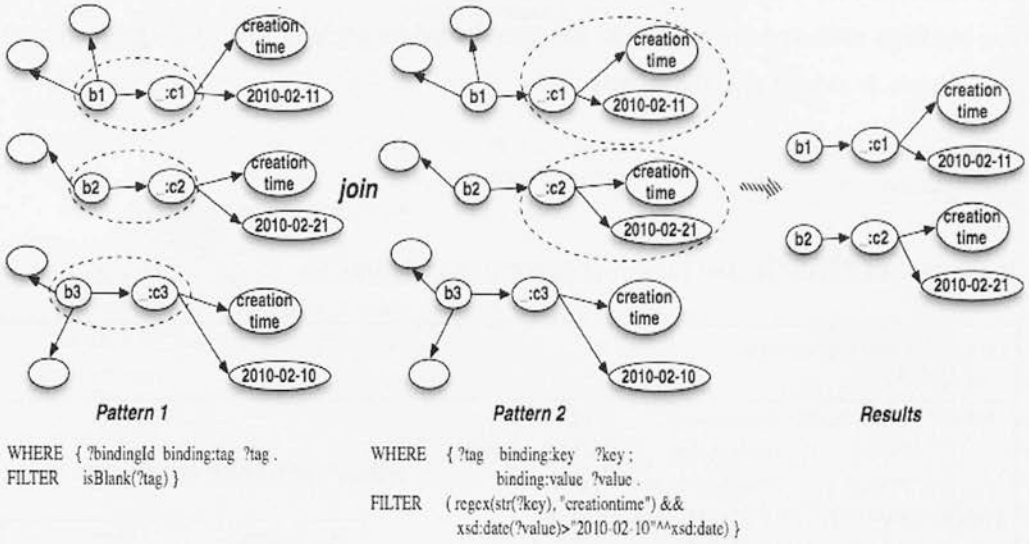


Figure 8.8: SPARQL multiple pattern matching is analogous to Inner-Join of SQL

In above queries, we have compared the two language in supporting various *string* related queries. Next, we examine the capabilities of the two query language in supporting statistical calculation and analysis.

Q8: "How many bindings has each user made in the last week?". For this question, a query should select the bindings made in the specified time duration, group them by the name of users, and count the numbers of members for each group. A SQL statement for these requirements becomes slightly complex:

```
SELECT t.TT_tag AS key, bst.BSTT_string_value AS value, COUNT(b.BT_bindingID) AS number
  FROM BindingTable b, TagTable t, Binding_StringType_KeyValuePairTagTable bst
 WHERE b.BT_bindingID = bst.BSTT_bindingID
    && bst.BSTT_tagID = t.TT_tagID
    && t.TT_tag = 'user'
    && b.BT_bindingID IN
          (SELECT b.BT_bindingID
             FROM BindingTable b, TagTable t, Binding_DataType_KeyValuePairTagTable bdt
            WHERE b.BT_bindingID = bdt.BDTT_bindingID
              && bdt.BDTT_tagID = t.TT_tagID
              && bdt.BDTT_date_value > '2010-02-14'
              && bdt.BDTT_date_value < '2010_02-22'
           )
GROUP BY bst.BSTT_string_value;
```

[15]SPARQL Language for RDF, http://www.w3.org/TR/rdf-sparql-query, retrieved on May 25, 2011.

The query joins BindingTable, TagTable and Binding_StringType_KeyValuePairTagTable to look up the user information, then group bindings by different users, and count the number in each group. An embedded `SELECT` statement (in the `WHERE` clause) includes the Binding_DateType_Key ValuePairTagTable to look up the date information, so as to ensure that the bindings retrieved are created in the specified time period. The SQL query, though complicate, is able to give the answer:

key	value	number
user	Alison	1

In the case of SPARQL, the following expression is desirable:

```
SELECT   COUNT(?bindingId)
  FROM  <binding.rdf>
 WHERE {{?bindingId  binding:tag     ?tag_0 .
         ?tag_0      binding:key     "creationtime" .
         ?tag_0      binding:value   ?value.
FILTER ((?value > "2010-02-14"^^xsd:date) &&
        (?value < "2010-02-22"^^xsd:date))  }
       { ?bindingId  binding:tag     ?tag_1.
         ?tag_1      binding:key     "user".
         ?tag_1      binding:value   ?user. }
}
GROUP BY ?user .
```

However, aggregation calculation (COUNT/AVERAGE/SUM/MAX/MIN) and associated machinery (GROUP BY, HAVING) are absent from W3C SPARQL specification[15]. Several implementations exist that would support such SPARQL queries even though they are outside the standard. Examples are: ARQ[16], ARC[17], and OpenLink Virtuoso[18]; they provide only limited aggregation capabilities. The features offered by different products are varied.

Q9: "How many times is each tag used?" This query asks about the tag frequency distribution.

The binding database table, TagTable, defines a field TT_stat to hold the tag frequency information. Therefore, a SQL answer for the question is straightforward, which simply retrieves the field value of TT_stat:

```
SELECT t.TT_tag AS tag, t.TT_stat AS number
FROME TagTable t;
```

[16] ARQ, A SPARQL Processor for Jena: http://jena.sourceforge.net/ARQ/

[17] ARC, Stand-alone SPARQL Parser: http://arc.semsol.org/

[18] OpenLink Virtuoso, commercial data server products, supporting RDF, SPARQL, and Linked Data processes: http://virtuoso.openlinksw.com/

However, even without TT_stat field, the question can still be answered by a SQL statement:

```
SELECT tag, COUNT(bindings) AS frequency FROM
  ((SELECT t.TT_tag AS tag, bt.BT_bindingID AS bindings
      FROM TagTable t, BindingTagTable bt
     WHERE t.TT_tagID = bt.BT_tagID
 UNION ALL
   (SELECT t.TT_tag AS tag, bdt.BT_bindingID AS bindings
      FROM TagTable t, Binding_DateType_KeyValuePairTagTable bdt
     WHERE t.TT_tagID = bdt.BDTT_tagID
 UNION ALL
   (SELECT t.TT_tag AS tag, bst.BT_bindingID AS bindings
      FROM TagTable t, Binding_StringType_KeyValuePairTagTable bst,
     WHERE t.TT_tagID = bst.BSTT_tagID))
x GROUP BY tag;
```

which obtains the following results:

tag	frequency
Isembryo029	1
disrodered	1
embryo	1
annotated by HDBR	1
head	1
creationtime	4
user	2

The query involves two processes: a) get all binding-tag pairs (from BindingTag tables and all Binding_(**datatype**)_KeyValuePairTagTables), and b) group them by tags and output the number of bindings in each group, that is the frequency of tag being used. A SPARQL query should be similar and the query string would be like this:

```
SELECT  ?tag_0  COUNT(?bindingId)
  FROM  <binding.rdf>
 WHERE { { ?bindingId  binding:tag     ?tag_0 .
           FILTER  (!IsBlank(?tag_0)) }
 UNION   { ?bindingId  binding:tag     ?tag .
           ?tag        binding:key     ?tag_0 . }}
GROUP BY ?tag_0
```

Process a) can be achieved: the query uses `UNION` to include both selections of binding-tag pairs and binding-key/value-tags pairs, which gives the following binding-tag pair set. However, without the supports of aggregation function, process b) cannot be performed.

bindingId	tag
13b2a8ee-dd76-47eb-962c-08f5430f313a	annotated by HDBR
43b2a8ee-dd76-47eb-962c-08f5430f313d	disrodered
73b2a8ee-dd76-47eb-962c-08f5430f313d	Isembryo029
13b2a8ee-dd76-47eb-962c-08f5430f313a	head
13b2a8ee-dd76-47eb-962c-08f5430f313a	embryo
13b2a8ee-dd76-47eb-962c-08f5430f313a	creationtime
13b2a8ee-dd76-47eb-962c-08f5430f313a	user
53b2a8ee-dd76-47eb-962c-08f5430f313a	creationtime
43b2a8ee-dd76-47eb-962c-08f5430f313d	user
43b2a8ee-dd76-47eb-962c-08f5430f313d	creationtime
73b2a8ee-dd76-47eb-962c-08f5430f313d	creationtime

Q10: "How many tags are used by each binding." This is another frequently used query which attempts to obtain the tag length distribution. A similar SQL query can be constructed as follows:

```
SELECT binding, COUNT(tag) AS tag_length FROM
  ((SELECT bt.BT_bindingID AS binding, t.TT_tag AS tag
      FROM TagTable t, BindingTagTable bt
     WHERE t.TT_tagID = bt.BT_tagID )
 UNION ALL
   (SELECT bdt.BDTT_bindingID AS binding, t.TT_tag AS tag
      FROM TagTable t, Binding_DateType_KeyValuePairTagTable bdt
     WHERE t.TT_tagID = bdt.BDTT_tagID )
 UNION ALL
   (SELECT bst.BSTT_bindingID AS binding, t.TT_tag AS tag
      FROM TagTable t, Binding_StringType_KeyValuePairTagTable bst,
     WHERE t.TT_tagID = bst.BSTT_tagID ))
x GROUP BY binding;
```

Similarly, the query firstly gathers all binding-tag pairs (from BindingTag tables and all Binding_(**datatype**)_KeyValuePairTagTables), and then groups them by bindings, and count the number of tags in each group, which is the tag length for each binding. The results are given below:

bindingId	tag_length
13b2a8ee-dd76-47eb-962c-08f5430f313a	5
43b2a8ee-dd76-47eb-962c-08f5430f313d	3
53b2a8ee-dd76-47eb-962c-08f5430f313a	1
73b2a8ee-dd76-47eb-962c-08f5430f313d	2

If `GROUP BY` and `COUNT` function could be supported, a SPARQL query would be as follows:

```
SELECT  ?bindingId  COUNT(?tag_0)
  FROM  <binding.rdf>
 WHERE { { ?bindingId  binding:tag     ?tag_0 .
           FILTER  (!IsBlank(?tag_0)) }
 UNION   { ?bindingId  binding:tag     ?tag .
           ?tag        binding:key     ?tag_0 . }
}
GROUP BY ?bindingId
```

Q11: "How many bindings are there with n tags for all n." To achieve this, an algorithm would involve three steps: a) retrieves all binding-tag pairs, which obtains dataset s_1; b) groups s_1 by bindings and count number of tags for each binding, which obtains dataset s_2; and c) group s_2 by tag-length and count the number of bindings for each distinguished tag number. SQL is able to handle this enquiry by adding another layer of `GROUP BY` clause, and the query statement is constructed below:

```
SELECT COUNT(binding), tag_length FROM
  (SELECT binding, COUNT(tag) AS tag_length FROM
     ((SELECT bt.BT_bindingID AS binding, t.TT_tag AS tag
        FROM TagTable t, BindingTagTable bt
       WHERE t.TT_tagID = bt.BT_tagID )
    UNION ALL
      (SELECT bdt.BDTT_bindingID AS binding, t.TT_tag AS tag
        FROM TagTable t, Binding_DateType_KeyValuePairTagTable bdt
       WHERE t.TT_tagID = bdt.BDTT_tagID )
    UNION ALL
      (SELECT bst.BSTT_bindingID AS binding, t.TT_tag AS tag
        FROM TagTable t, Binding_StringType_KeyValuePairTagTable bst,
       WHERE t.TT_tagID = bst.BSTT_tagID ))
   x GROUP BY binding; )
x GROUP BY tag_length
```

That is, an additional aggregation calculation is applied to the result datasets of query **Q10**, which gives the answer:

binding_number	tag_length
1	5
1	3
1	1
1	2

However, a single SPARQL query would be insufficient for the task. It would be useful for SPARQL to provide pipe operations which deliver the outputs of the first query as the input for a second query.

The pressure to extend the language is evident. The SPARQL Extension Working Group[19] (and some other groups) have identified a list of requirements for extension, including but not limited to:

- To provide INSERT, UPDATE, DELETE operations;
- To provide aggregation functions and the associated machinery;
- To provide sub-query which allows adding SELECT statements within WHERE;
- To provide user defined functions;
- To provide functions of finding and matching structured paths of arbitrary lengths between RDF nodes;
- To provide federated query, approaches to federating SPARQL end-points.

[19] http://esw.w3.org/topic/SPARQL/Extensions lists all proposed topics. Retrieved on March 2010.

Despite the fact that improvements efforts are needed, it is undeniable that SPARQL is a potential powerful query language which provides simple pattern matching approach to navigating an RDF graph. Together with carefully constructed RDF graphs, SPARQL is capable of answering many (7 out of 11) desired binding inquires.

However, SPARQL seems unsuitable for intensive statistical calculations which would be essential for binding operations. Depending on characteristics of data and their potential usage, suitable data model and query language should be chosen. Regarding the simple representation of binding which results in a shallow RDF graph, there are no practical advantages to use the RDF data model, nor SPARQL for query, except that they are endorsed as W3C standards. SQL, which is designed for relational data model providing efficient filter and calculation facilities, seems still the ideal choice.

As a brief conclusion, the forth constraint is induced:

Constraint Rule 4: Binding query capabilities should be sufficient and efficient for binding enquires.

8.6 Comparison of Communication Capabilities

The binding service provides communication operations to get data/metadata or their status through APIs:

⇒ GetResource(*ref*: URI)
⇒ GetResourceStatus(*ref*: URI)

The implementation supports numerous network protocols, such as HTTP, FTP, and GridFTP, through OGSA-DAI service activities. The Linked Data Model mainly uses HTTP (and SOAP) transport protocol for dealing with SPARQL queries. In the case of scientific data, the requirements for data size, security, format and network performance are different, and can go beyond the capabilities of HTTP (and SOAP). For example, there are clear needs in e-Science to handle large-scale flat format of data [Kett 08]. Currently, there are active researches on using Hadoop/Cloud computing to parallelise SPARQL queries. In summary, the fifth constraint is given as:

Constraint Rule 5: Transport protocols should be able to support the requirements emerged in scientific data that enable secure, reliable and high performance data movement.

It is rather like an extension requirement for the Linked Data. It is regarded as a constraint-rule in the sense it restricts the quality and performance of data delivery to be higher level.

8.7 Summary

This section developed a detailed comparison of the Linked Data approach with the binding service approach, and Table 8.1 provides a summary.

Table 8.1: Summary of the Comparisons

	The Linked Data Approach	**The Binding Service Approach**
Data Structure	$< subject, predicate, object >$	$< binding\_id, ref\_sub, ref\_obj, tags >$
Binding Expression	Structured RDF graph	$< binding\_id, ref\_sub, ref\_obj, tags >$
Binding Storage	Loosely-coupled information base	Transaction-based information store
Search	Support SPARQL query	Support SQL query
Communication	HTTP. No support for high-performance nor other types of network protocols	Through OGSA-DAI interface which supports HTTP, FTP, GridFTP protocols

We have compared the data structure of the two approaches, and indicated that the binding data structure was different from the RDF data model. Importantly, binding defines globally unique identifier makes it possible for supports of data integration and identification in distributed systems. For RDF records to have identifiable RDF triples, technology such as Named Graph should be applied. On the other hand, RDF is an expressive data model. By applying structures (defining binding patterns, adding namespaces and specifying data types), bindings can be expressed in RDF graphs.

We have compared data storage mechanism of an RDF binding store and a binding database, and indicated that the RDF binding store was not transaction-based. Transaction control maintains the persistent states of a storage system, supports concurrent processes and ensures unexpected failures can be handled. A lack of such transaction management can easily cause information inconsistency in a storage system.

We have compared the query capability of SQL (used by the binding service), and SPARQL (used by Linked Data approaches), and indicated that SQL was more suitable for scientific applications which require intensive statistical processes. SPARQL is a powerful language, however, extensions are needed to provide more desired functions.

Finally, we have compared the communication capabilities of the two approach. The Linked Data approach uses HTTP (and SOAP) protocols for dealing with SPARQL queries, while the prototype implementation of binding service exposes a OGSA-DAI interface which supports HTTP, FTP, and GridFTP. As the requirements of scientific applications for data scale, security, format and network performance could go beyond the capability of HTTP and (SOAP), reliable and high performance network protocols should be supported.

In conclusion, the binding approach is a different approach from the Linked Data model. The differences have been largely determined by the requirements comes from the computational contexts. The binding service is designed for managing scientific data and

metadata, and aimed to address the new challenges posed by the newly emerged phenomena.

Chapter 9

Conclusions and Future Work

This chapter reviews the steps that have been taken to investigate the thesis hypothesis and the findings that result from the research. It also discusses aspects for improvements and areas for future explorations.

The chapter is organised in two sections: section 8.1 concludes the thesis; and section 8.2 discusses the future work.

9.1 Conclusions of the Thesis

The goal of the thesis was to investigate the hypothesis that simple binding system, which stored and manipulated the binding representations between data and metadata was both feasible and useful, in combination with other services and tools, for serving the various types and scales of scientific data in a distributed computing context. We review the research processes to achieve this goal below.

The investigation began from an observation of the scientific data and metadata phenomena. We have found that the scientific data (and metadata) presented two remarkable characteristics, 1) large in scale and 2) diverse in representation, introducing significant challenges for management. Although binding is a meaningful notion, its concept had not been well established in this context. To develop an understanding of bindings, we used a binding classification framework to examine five classes of metadata centric scientific applications. We have found that binding management was an exploratory field with many and varied applications, and the widespread need to support the creation and use of bindings was clear. To identify binding related problems, we scoped the study into a typical domain application. The empirical analysis of the EurExpress system provided evidence that bindings were vulnerable to failures in the processes that created and maintained them, and to failures in the systems that stored their representation. Without effective management, the information inconsistency would grow and propagate widely.

Consequently, the expensively generated data would lose their value, and the analysis and research based on them would be untrustworthy.

With an improved understanding of bindings, we have explored existing approaches and related technologies. Firstly, we reviewed the scientific applications dealing with data and metadata, and examined them from the binding point of view. We noted that most existing systems implemented the *dependent* binding management approach which co-located the metadata and/or data in centralised repository. In contrast, the OntoGrid system demonstrates the advantage of using the *independent* binding management approach, that independently managing bindings no longer requires data and/or metadata to be copied into a central server, and the information would be up-to-date with the original data sources. It also does not require co-operation from metadata and data owners, except for stability of references. In addition, it could list metadata and data held in multiple sources. Unfortunately, the OntoGrid approach was not a generic data-metadata management approach. The second area we have explored was data-reference approaches, because an *independent* binding model uses data-reference methods to reference data held in remote sources. Three state-of-the-art systems have been analysed. The conclusions were: 1) Web 2.0 mashup provides only basic data-reference functionality; 2) The iRODS did not support storage systems beyond its control (which was part of binding requirements); and 3) The Linked Data model addressed similar issues as the binding approach. Finally, we explored the related technologies that could be adopted to develop a binding system. We have found that tagging was an efficient and flexible annotation technology and suitable for large and heterogeneous data discovery, while the Cloud could provide simple solutions for system scalability.

Next, we used a formal model to capture the binding characteristics and management requirements obtained from both observations of binding applications and analysis of the existing approaches. The formal model has established the formal foundation of the binding-oriented approach where bindings were the first-class citizens and had an existence independent of their associated data and metadata. The binding basic model defined the binding data structure and behavioural operations. Extending the basic model, we have defined the consistent binding model and formalised the binding inconsistency problems raised in the analysis of the EurExpress system.

Although the formal model has described system properties a binding system should comprise, it has left a number of challenges for design. In the design process, we proposed the computational representation of a binding, which used UUID to represent the global unique binding *identifier*, used URIs to represent the *subject* and the *object* that are the references of data and metadata resources, respectively, and used tags to describe the binding. A set of operations for the binding are specified based on the formal definitions.

To evaluate the viability of the design, a prototype binding service has been implemented. The design of the binding service is kept simple which can largely reduce implementation costs and complexity when integrating with an external application. The binding service is a generic tool which can be integrated with various applications. An existing system can adopt a binding service to have binding facilities automatically, i.e. to create and look up bindings, and to validate consistency of information.

The performance evaluation of the binding service has been obtained by experimental analysis. A number of existing scientific systems were observed and analysed to characterise the workloads that a binding service would need to support. These were then fitted to parametric statistical description of the workloads so that synthetic workloads with comparable properties could be simulated. The simulated workloads were applied to the binding service to measure performance. The performance results show that the binding service is capable of handling different types of workloads in small to large scales. In the subsequent experiments, we drilled down to the scalability issue. The results of the experimental evaluation of the Hadoop-based binding storage show that with merely basic configurations, the Hadoop binding system provide scalable storage capacities (sufficient for the current investigation), and Hadoop is potentially useful to serve as a simple strategy for scaling up the binding system.

To prove the value of the binding approach and demonstrate binding usage, we have provided a sample installation of a binding service to the Testing EurExpress system, and the experimental evaluation results. The results show that the binding service is able to detect consistency failures embedded in the scientific data at affordable computational cost. To illustrate the binding usage in other types of scientic applications, the binding solutions to seven representative use scenarios have been provided. The results show that the binding service is a generic tool which can be easily integrated with other services or software tools and support various use cases.

Finally, we have examined the previous question that whether the Linked Data approach was identical to the binding approach and was more advanced to serve the binding requirements? We have compared the two approaches from five perspectives, and the discoveries influenced our decisions included: 1) an RDF graph would need to adopt an agreed structure and ontology so that the mapping to and from binding was consistently interpreted; 2) an RDF binding store would not be transaction-based; 3) the SPARQL query language would need extensions to support statistical functions, and SQL would be more suitable to process scientific data; and 4) the Linked Data approach used only HTTP network protocol which would be insufficient for supports of scientific data and metadata. The conclusions were that the binding approach was a different approach to the data-reference model. The design of the simple binding service was more feasible to the

requirements for scientific data and metadata. Therefore, we proceeded with the original design.

As the final remark, Jim Gray (2007) indicated that currently scientists have very few generic tools for collecting and analysing and processing the data; and only a small number of scientific projects can afford expensive software development and scalable repositories, and a large number of small projects cannot have the facilities [Gray 07]. The proposal of the binding approach is a response to this important call. By providing computational capabilities over references to scientific data/metadata (rather than hosting them), the binding approach enables a generic and cheap solution for managing large-scale heterogeneous scientific data and metadata. The approach targets the users of those small scientific projects which are in a large number, to help them deliver high quality of research in a cheaper way.

The study is timely. In March 2009 after the web being invented for 20 years, Tim Berners-Lee initiated the open data movement. Towards the end of the year, both UK and US governments make the data open accessible under the principles of the Linked Data. This study looks at one of specific domains of the open data, the scientific data and metadata. Linking scientific data allows scientists to reference results of other experiments or studies from other domain areas to gain insights of the knowledge. In a sense, it enables science to be conducted through more open and scalable collaborative activities. Although the focus was on the data and metadata, the approach has the capacity to transparently include and interrelate resources and other relevant information.

9.2 Future Work

There are several elements could be extended in future work.

Firstly, in Chapter 5, we have used only basic configurations to test the Hadoop-based binding storage. This was because the experimental equipment was limited when we performed the experiment. Although the experiment exercises have provided us with useful experience and sufficient results for current investigation, once the resource is available, we plan to use well-turned Hadoop clusters, which would provide more realistic and larger scale measures.

Secondly, the comparison of the binding service and the Linked Data approach in Chapter 8 has been focused on the system properties. This was because the main purpose of the comparison was to evaluate the feasibility of the design. Although we have obtained sufficient evidence to make the decision, it would be more convincing if the performance measures could also be provided. We plan to implement a Linked-Data-based binding service and use the controlled experiment to compare the approaches. For example, we

could use the workload simulator to generate various types of workloads at various scales, and simultaneously apply them to the two services. The performance of response time and throughput can then be collected for comparisons.

Finally, in Chapter 7, for the demonstration purposes we have provided realisations of two types binding inconsistency checking by the prototype implementation of the constraint language. In future research, we can extend the implementation to address different types of binding failures. The results can provide more evidence to show the usefulness of the binding service.

The study also opens up numerous directions for future research.

We believe that the binding service is potentially useful to help individual scientists with their work, and we will encourage the use of this technology. We hope to introduce the binding service to domain scientists, help them to install the system, and provide guidance for use. We will use surveys to collect user feedback, and monitor workloads to analyse the usability. Both will be used for continuing improvements of the design and implementation.

Another direction could be workload modelling and simulation for scientific data. During the study we found that many scientific systems did not provide workloads for analysis, which probably because the collection of such information would require extra implementation and management effort. We have analysed workloads of a number of existing scientific systems and shown that different systems could have very different workload patterns. As different workload inputs may lead to very different performance results, it is important to have workload evidence for decision making in design and implementation. For a future task, we will look at more scientific data and metadata systems and identify new characteristic workload patterns. We then can extend the simulator to generate new types of workloads, which can be used for later evaluations or to support other researchers.

Finally, the positive experience with Hadoop motivates further explorations in the area of the Cloud computing. We have noted the potential to enhance the performance of the Hadoop, and we propose a novel approach for optimising binding processes over Hadoop clusters. Briefly, we have identified that data partitioning was crucial to the performance of binding creation and retrieval over a cluster, and we propose two intelligent partitioning algorithms, K-Means Clustering and Fuzzy K-Means Clustering, which assign bindings into Hadoop *datanode*s so that bindings in the same node are similar, where the similarity is measured by the *distance* between the binding and the *centroid* of *clusters* formed by K-Means Clustering algorithm. The hypothesis is, if similar bindings are stored together, data locality will be high, that is if a binding satisfies a query, it is likely that the 'friends' of the binding will as well. A Fuzzy K-Means Clustering is to allow overlap of *clusters*.

This is to examine whether a certain level of redundancy could improve or degrade the system performance. Appendix B describes the algorithms in details, and provides an experiment plan for testing the hypothesis.

In the longer term, we envisage binding services as part of the web-service ecosystem being widely used by researchers and other applications. The APIs would evolve to accommodate usage but continue to support the primitive functions introduced in this thesis. New engineering approaches would then be justified, such as incremental rule validation on update or periodically. Investment in tools that use the binding services would make them easy and intuitive to use. How best to present the binding facilities in those tools is an open research question.

Bibliography

[Adam 02] L. A. Adamic and B. A. Huberman. "Zipf's law and the Internet". *Glottometrics*, Vol. 2, pp. 143–150, 2002.

[Ahma 10] Y. Ahmad, R. Burns, M. Kazhdan, C. Meneveau, A. S. Szalay, and A. Terzis. "Scientific Data Management at the Johns Hopkins Institute for Data Intensive Engineering and Science". *ACM SIGMOD Record*, Vol. 39, No. 3, 2010.

[Aitk 07a] S. Aitken and Y. Chen. *Anatomy Ontologies for Bioinformatics: Principles and Practice*, Chap. COBrA and COBrA-CT:Ontology Engineering Tools. Springer, 2007.

[Aitk 07b] S. Aitken and Y. Chen. "Managing the transition from OBO to OWL: The COBrA-CT Bio-Ontology Tools". In: *the 2007 UK e-Science All Hands Meeting*, 2007.

[Aitk 08] S. Aitken, Y. Chen, B. Webber, W. F. Fan, and J. Bard. "OBO Explorer: An Editor for Open Biomedical Ontologies in OWL". *Oxford Journals: Bioinformatics*, Vol. 24(3), pp. 443–44, December 2008.

[Alpe 06] P. Alper, O. Corcho, I. Kotsiopoulos, P. Missier, S. Bechhofer, D. Kuo, and C. Goble. "S-OGSA as a Reference Architecture for OntoGrid and for the Semantic Grid". In: *3rd GGF Semantic Grid Workshop, GGF16*, Feb. 2006.

[Alti 04] I. Altintas, C. Berkley, E. Jaeger, M. Jones, B. Ludascher, and S. Mock. "Kepler: An Extensible System for Design and Execution of Scientific Workflows". In: *Proceedings. 16th International Conference on Scientific and Statistical Database Management*, pp. 423–424, Jun. 2004.

[Alti 05] I. Altintas, A. Birnbaum, K. Baldridge, W. Sudholt, M. Miller, C. Amoreira, Y. Potier, and B. Ludascher. "Framework for the Design and Reuse of Grid Workflows". In: *Workshop on Scientific Applications on Grid Computing (SAG'04)*, 2005.

[Alti 06] I. Altintas, O. Barney, and E. Jaeger-Frank. "Provenance Collection Support in he Kepler Scientific Workflow System". In: *the Proceeding of 2006 International Provenance and Annotation Workshop (IPAW)*, 2006.

[Anto 07] M. Antonioletti, N. P. Chue Hong, A. C. Hume, M. Jackson, K. Karasavvas, A. Krause, J. M. Schoof, P. M. Atkinson, B. Dobrzelecki, M. Illingworth, N. McDonnell, M. Parsons, and E. Theocharopoulos. "OGSA-DAI 3.0 –The Whats and the Whys". In: *the 2007 UK e-Science All Hands Meeting*, 2007.

[Anto 09] M. Antonioletti, C. B. Aranda, O. Corcho, M. Esteban-Gutierrez, A. Gomez-Perez, I. Kojima, S. Lynden, and S. M. Pahlevi. "WS-DAI RDF(S) Realiation: Introduction, Motivational Use Cases and Terminologies". Tech. Rep. GFD-I.163, Open Grid Forum DAIS Working Group, 2009.

[Armb 09] M. Armbrust, F. A., R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. "Above the Clouds: A Berkeley View of Cloud Computing". *EETimes techonline*, 2009.

[Asta 05] V. Astakhov, A. Gupta, S. Santini, and J. S. Grethe. "Data Integration in the Biomedical Informatics Research Network (BIRN)". In: *the Processing of Second International Workshop at the Data Integration in Life Sciences*, 2005.

[Asta 06] V. Astakhov, A. Gupta, J. S. Grethe, E. Ross, D. Little, A. Yilmaz, X. Qian, S. Santini, M. Martone, and M. Ellisman. "Semantically Based Data Integration Environment for Biomedical Research". In: *the 19th IEEE International Symposium on Computer-Based Medical Systems*, 2006.

[Atki 09] P. M. Atkinson and D. De Roure. "Data-Intensive Research: making best use of research data". Tech. Rep., National eScience Centre, 2009.

[Atki 89] P. M. Atkinson, F. Bancilhon, D. DeWitt, K. Dittrich, D. Maier, and S. Zdonik. "The Object-Oriented Database System Manifesto". In: *the Proceeding of the First International Conference on Deductive and Object-Oriented Database*, 1989.

[Auer 09] S. Auer, S. Dietzold, J. Lehmann, S. Hellmann, and D. Aumueller. "Triplify – Light-Weight Linked Data Publication from Relational Databases". In: *18th World Wide Web Conference*, 2009.

[Auth 09] T. I. D. R. W. Authors. "Prepublication data sharing". *Nature*, No. 461, pp. 168–170, Sep. 2009.

[Baba 93] O. Babaoglu and K. Marzullo. "Consistent Global States of Distributed Systems: Fundamental Concepts and Mechanisms". Tech. Rep., Laboratory for Computer Science, University of Bologna, 1993.

[Baez 08] R. Baeza-Yates and R. Ramakrishnan. "Data Challenges at Yahoo!". In: *Proceedings of the 11th International Conference on Extending database technology: Advances in database technology*, 2008.

[Bala 06] N. Balakrishnan, Ed. *Encyclopedia of statistical sciences*. Hoboken, N.J. : Wiley-Interscience, c2006., 2nd Ed., 2006.

[Beag 06] N. Beagrie. "Digital Curation for Science, Digital Libraries and Individuals". *The International Journal of Digital Curation*, Vol. 1, No. 1, 2006.

[Beck 08] C. Becker and C. Bizer. "DBpedia Mobile – A Location-Aware Semantic Web Client". In: *Proceeding of the Semantic Web Challenge at ISWC 2008*, 2008.

[Beck 09] C. Becker and C. Bizer. "Exploring the Geospatial Semantic Web with DBpedia Mobile". In: *Web Semantics: Science, Services and Agents on the World Wide Web*, pp. 278–286, December 2009.

[Bege 11] K. Begeman, A. N. Belikov, D. R. Boxhoorn, F. Dijkstra, H. Holties, Z. Meyer-Zhao, G. A. Renting, E. A. Valentijn, and W. J. Vriend. "LOFAR Information System". *Future Generation Computer Systems*, Vol. 27, pp. 319–328, 2011.

[Bell 02] W. H. Bell, D. G. Cameron, L. Capozza, A. P. Millar, K. Stockinger, and F. Zini. "Simulation of Dynamic Grid Replication Strategies in OptorSim". In: *The Proceeding of the 3rd International Workshop on Grid Computing*, 2002.

[Bene 98] D. Beneventano, S. Bergamaschi, S. Lodi, and C. Sartori. "Consistency Checking in Complex Object Database Schemata with Integrity Constraints". In: *IEEE Transactions On Knowledge and Data Engineering*, 1998.

[Benn 09] S. Bennett, M. Bhuller, and R. Covington. "Architectural Strategies for Cloud Computing". In: *Oracle White Paper in Enterprise Architecture*, Aug. 2009.

[Bens 08] D. A. Benson, I. Karsch-Mizrachi, D. J. Lipman, J. Ostell, and D. L. Wheeler. "GenBank". *Nucleic Acids Research*, Vol. 36 Database issue, pp. D25–30, 2008.

[Berm 02] H. M. Berman, T. Battistuz, T. N. Bhat, W. F. Bluhm, P. E. Bourne, K. Burkhardt, Z. Feng, G. L. Gilliland, L. Iype, S. Jain, P. Fagan, J. Marvin, D. Padilla, V. Ravichandran, B. Schneidar, N. Thanki, H. Weissig, J. D. Westbrook, and C. Zardecki. "The Protein Data Bank". *Biological Crystallography*, Vol. Acta Crystallographic Section D, No. ISSN 0907-4449, 2002.

[Bern 00] P. A. Bernstein, A. Y. Halevy, and R. A. Pottinger. "A Vision for Management of Complex Models". *SIGMOD Record*, Vol. 29, No. 4, 2000.

[Bern 05] T. Berners-Lee, R. T. Fielding, and L. Masinter. "Uniform Resource Identifier (URI):Generic Syntax". Standards, IETF, Jan. 2005.

[Bern 06] T. Berners-Lee. "Linked Data – W3C Design Issues: Architectural and Philosophical Points". Tech. Rep., www.w3.org/DesignIssues/LinkedData.html, 2006.

[Bert 96] E. Bertino, E. Ferrari, and G. Guerrini. "A Formal Temporal Object-Oriented Data Model". In: *the Proceeding of 5th International Conference on Extending Database Technology: Advances in Database Technology*, 1996.

[Bezd 81] J. C. Bezdek. *Pattern Recognition With Fuzzy Objective Function Algorithms*. Kluwer Academic Publishers Norwell, MA, USA, 1981.

[Bhag 10] J. Bhagat, F. Tanoh, E. Nzuobontane, T. Laurent, J. Orlowski, M. Roos, K. Wolstencroft, S. Aleksejevs, R. Stevens, S. Pettifer, R. Lopez, and C. Goble. "BioCatalogue: a universal catalogue of web services for the life sciences". *Nucleic Acids Research*, No. doi: 10.1093/nar/gkq394, 2010.

[Bize 06] C. Bizer and R. Cyganiak. "D2R Server – Publishing Relational Databases on the Semantic Web". In: *5th International Semantic Web Conference (ISWC2006)*, 2006.

[Bize 08] C. Bizer and A. Schultz. "Benchmarking the Performance of Storage Systems that expose SPARQL Endpoints". In: *the 4th International Workshop on Scalable Semantic Web knowledge Base Systems*, 2008.

[Bize 09a] C. Bizer and R. Cyganiak. "Quality-driven Information Filtering using the WIQA Policy Framework". *Journal of Web Semantics*, Vol. 7(1), pp. 1–10, 2009.

[Bize 09b] C. Bizer, T. Heath, and T. Berners-Lee. "Linked Data – The Story So Far". *Special Issue on Linked Data, International Journal on Semantic Web and Information Systems (IJSWIS)*, 2009.

[Bize 09c] C. Bizer and A. Schultz. "The Berlin SPARQL Benchmark". In: *International Journal On Semantic Web and Information Systems: Special Issue on Scalability and Performance of Semantic Web Systems*, 2009.

[Blan 09] C. V. Blanco, E. Huedo, R. S. Montero, and I. M. Liorente. "Dynamic Provision of Computing Resources From Grid Infrastructures and Cloud Providers". In: *the Grid and pervasive Computing Conference*, 2009.

[Blei 08] J. Bleiholder and F. Naumann. "Data Fusion". *ACM Computing Surveys*, Vol. 41(1), pp. 1–41, 2008.

[Boha 06] P. Bohannon, E. Elnahrawy, W. Fan, and M. Flaster. "Putting Context into Schema Matching". In: *the Proceeding of the 32nd International Conference on VLDB*, 2006.

[Bola 06] G. A. Bolado and G. Eichele. "Analysing the developing brain transcritome with the GenePaint platform". *The Journal of Physiology*, Vol. 575, pp. 347–352, 2006.

[Bose 06a] R. Bose, P. Buneman, and D. Ecklund. "Annotating scientific data: why it is important and why it is difficult". In: *the 2006 UK e-Science All Hands Meeting*, 2006.

[Bose 06b] R. Bose, R. G. Mann, and D. Prina-Ricotti. "AstroDAS: Sharing Assertions Across Astronomy Catalogues Through Distributed Annotation". In: *the 2006 International Provenance and Annotation Workshop (IPAW)*, 2006.

[Bosn 02] A. Bosneag and M. Brockmeyer. "A FORMAL MODEL FOR EVENTUAL CONSISTENCY SEMANTICS". *International Conference on Parallel and Distributed Computing Systems, PDCS 2002*, pp. 204–209, 2002.

[Brei 92] Y. Breitbart, H. G. Molina, and A. Silberschatz. "Overview of Multidatabase Transaction Management". *VLDB Journal*, Vol. 2, pp. 181–239, 1992.

[Bruh 04] I. Bruha. "Meta-Learner for Unknown Attribute Values Processing: Dealing with Inconsistency of Meta-Databases". *Journal of Intelligent Information Systems*, Vol. 22, No. 1, 2004.

[Buyy 08] R. Buyya, C. S. Yeo, and S. Venugopal. "Market-Oriented Cloud Computing: Vision, Hype and Reality for Delivering IT Services as Computing Utilities". In: *the Proceedings of the 10th IEEE International Conference on High Performance Computing and Communications (HPCC 2008)*, Sep. 2008.

[Camp 09] R. Campbell, I. Gupta, M. Heath, S. Y. Ko, M. Kozuch, M. Kunze, T. Kwan, K. Lai, H. Y. Lee, M. Lyons, D. Milojicic, D. O'Hallaron, and Y. C. Soh. "Open CirrusTM Cloud Computing Testbed: Federated Data Centers For Open Source Systems and Services Research". In: *Workshop on Hot Topics in Cloud Computing (HotCloud'09)*, Jun. 2009.

[Carr 03] J. J. Carroll and P. Stickler. "RDF Triples in XML". Tech. Rep., HP Labs, 2003.

[Carr 05] J. J. Carroll and C. Bizer. "Named Graphs, Provenance and Trust". In: *WWW 2005*, 2005.

[Cast 09] P. Castagna, A. Seaborne, and C. Dollin. "A Parallel Processing Framework for RDF Design and Issues". In: *HP Laboratories Technical Reports*, Oct. 2009.

[Ceri 93] S. Ceri and J. Widom. "Managing Semantic Heterogeneity with Production Rules and Persistent Queues". In: *the Proceedings of 19th VLDB Conference*, 1993.

[Chan 06] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. "BigTable: A Distributed Storage System for Structured Data". In: *the 7th USENIX Symposium on Operation Systems Design and Implementation, OSDI'06*, 2006.

[Char 02] S. C. Charleston. "Self-stabilization and eventual consistency in replicated real-time database". In: *the Proceeding of the first workshop on self-healing systems*, 2002.

[Chen 07] Y. Chen, D. Berry, and P. Dantressangle. "Transaction-Based Grid Database Replication". In: *the 2007 UK e-Science All Hands Conference*, 2007.

[Cher 05] A. Chervenak, R. Schuler, C. Kesselman, S. Koranda, and B. Moe. "Wide Area Data Replication for Scientific Collaborations". In: *the 6th IEEE ACM International Workshop on Grid Computing*, 2005.

[Chi 08] E. H. Chi and T. Mytkowicz. "Understanding the Efficiency of Social Tagging Systems using Information Theory". In: *the Preceding of 2008 ACM Hypertext Conference*, 2008.

[Cial 01] R. Cialdini. *Influence: Science and Practices*. Allyn and Bacon, 2001.

[Cicc 04] A. J. Ciccone and B. Hamel. "The Next Generation: Q Replication". *DB2 Magazine*, Vol. 9, No. 3, 2004.

[Cigi 05] M. Cigian and L. Hluchy. "Towards Scalable Grid Replica Optimization Framework". In: *the Proceeding of the 4th International Symposium on Parallel and Distributed Computing*, 2005.

[Clem 10] M. Clements, A. P. de Vries, and M. J. T. Reinders. "The Influence of Personalisation on Tag Query Length in Social Media Search". *Information Processing and Management*, Vol. 46, pp. 403–412, 2010.

[Clif 95] C. B. Clifford. "Sybase Replication Server Primer (McGraw-Hill Computer Communications Series)". *Mcgraw-Hill(Tx)*, 1995.

[Coet 08] P. Coetzee, T. Heath, and E. Motta. "SparqPlug: Generating Linked Data from Legacy HTML, SPATQL and the DOM". In: *1st Workshop on Linked Data on the Web (LDOW2008)*, 2008.

[Corc 06] O. Corcho, P. Alper, I. Kotsiopoulos, P. Missier, S. Bechhofer, and C. Goble. "An Overview of S-OGSA: A Reference Semantic Grid Architecture". In: *Web Semantics: Science, Services and Agents on the World Wide Web In Semantic Grid – The Convergence of Technologies*, pp. 102–115, 2006.

[Corc 07] O. Corcho, P. Alper, P. Missier, S. Bechhofer, C. Goble, and W. Xing. "Metadata Management in S-OGSA". *Computational Science – ICCS 2007*, 2007.

[Corn 04] D. R. Corney, M. de Vere, T. Folkes, D. Giaretta, K. Kleese van Dam, B. N. Lawrence, S. J. Pepler, and B. Strong. "Applying the OAIS standard to CCLRC's British Atmospheric Data Centre and the Atlas Petabyte Storage Service". In: *the 2004 UK e-Science Programme All Hands Meeting*, 2004.

[Cudr 08] P. Cudre-Mauroux, A. Budura, M. Hauswirth, and K. Aberer. "PicShark: Mitigating Metadata Scarcity Through Large- Scale P2P Collaboration". *The VLDB Journal*, Vol. 17, pp. 1371–1384, 2008.

[Cumm 07] G. Cumming, F. Fidler, and D. L. Vaux. "Error Bars in Experimental Biology". *The Journal of Cell Biology*, Vol. 177, No. 1, Apr. 2007.

[Curc 08] V. Curcin and M. Ghanem. "Scientific workflow systems - can one size fit all?". In: *Biomedical Engineering Conference, 2008. CIBEC 2008. Cairo International*, pp. 1–9, Dec. 2008.

[Dean 04] J. Dean and S. Ghemawat. "MapReduce: Simplified Data Processing on Large Clusters". In: *6th Symposium on Operation Systems Design and Implementation OSDI'04*, 2004.

[Deel 09] E. Deelman, D. Gannon, M. Shield, and I. Taylor. "Workflows and e-Science: An overview of workflow system features and capabilities". *Future Generation Computer Systems*, Vol. 25, pp. 528–540, 2009.

[Dell 08] K. Dellschaft and S. Staab. "An Epistemic Dynamic Model for Tagging Systems". In: *the Preceding of 2008 ACM Hypertext Conference*, 2008.

[Demb 03] D. Dembele and P. Kastner. "Fuzzy C-Means Method for Clustering Microarray Data". *Bioinformatics*, Vol. 19, No. 8, pp. 973–980, 2003.

[DeWi 90] D. J. DeWitt, S. Ghandeharizaseh, D. Schneider, A. Bricker, H. I. Hsiao, and R. Rasmussen. "The Gamma Database Machine Project". *IEEE Transactions on Knowledge and Data Engineering*, 1990.

[Dobr 10] B. Dobrzelecki, A. Krause, A. Hume, A. Grant, M. Antonioletti, T. Alemu, M. Atkinson, M. Jackson, and E. Theocharopoulos. "Integrating distributed data sources with OGSA–DAI DQP and Views". *Phil. Trans. R. Soc. A*, Vol. 368, No. 1926, pp. 4133–4145, 2010.

[Dunl 08] L. Dunlap. "Advancing Gene Expression Studies". *Genetic Engineering and Biotechnology News*, Vol. 28, No. 14, Aug. 2008.

[Elli 03] M. Ellisman and S. Peltier. "Medical Data Federation: The Biomedical Informatics Research Network". In: I. Forster and C. Kesselman, Eds., *The Grid 2: Blueprint for a New Computing Infrastructure*, Chap. 8, Morgan Kaufmann, Dec. 2003.

[Evan 98] A. Evans, R. France, K. Lano, and B. Bumpe. "The UML as a Formal Modelling Notation". In: *the Proceeding of the 1st International Workshop on the Unified Modelling Language ¡¡UML¿¿'98: Beyond the Notation*, 1998.

[Fan 09a] W. Fan, F. Geerts, L. V. S. Lakshmanan, and M. Xiong. "Discovering Conditinal Functional Dependencies". In: *the Proceeding of 25th International Conference on Data Engineering*, 2009.

[Fan 09b] W. Fan, X. Jia, J. Li, and S. Ma. "Reasoning about Record Matching Rules". In: *the Proceeding of the VLDB Endowment*, 2009.

[Feit 02] D. G. Feitelson. "Workload Modelling for Computer Systems Performance Evaluation". *Performance Evaluation of Complex Systems: Techniques and Tools*, Vol. 2459/2002, pp. 114–141, 2002.

[Fern 00] M. Fernandez, D. Florescu, A. Levy, and D. Suciu. "Declarative specification of Web sites with STRUDEL". In: *the VLDB Journal*, pp. 38–55, 2000.

[Fern 08] E. Fernandez. "Biosemiotics and Self-reference from Peirce to Rosen". In: *the Proceeding of the 8th Annual International Gatherings in Biosemiotics*, 2008.

[Fern 99] M. Fernandez, D. Florescu, A. Levy, and D. Suciu. "Verifying Integrity Constraints on Web Sites". In: *the Proceeding of the 16th international joint conference on Artificial Intelligence*, 1999.

[Fost 02] I. Foster, J. Vockler, M. Wilde, and Y. Zhao. "Chimera: A Virtual Data System for Representing, Querying and Automating Data Derivation". In: *the Proceeding of the 14th Conference on Scientific and Statistical Database Management*, 2002.

[Frew 01] J. Frew and R. Bose. "Earth System Science Workbench: A Data Management Infrastructure for Earth Science Products". In: *the Proceeding of the 13th International Conference on Scientific and Statistical Database Management*, 2001.

[Garm 03] J. Garmany and R. G. Freeman. *Oracle Replication: Snapshot, Multi-master and Materialized Views Scripts (Oracle In-Focus series).* Rampant Techpress, 2003.

[Ge 08] T. Ge, S. Zdonik, and S. R. Madden. "Top-K Queries on Uncertain Data: On Score Distribution and Typical Answers". In: *SIGMOD*, 2008.

[Geha 10] A. Gehani, M. Kim, and T. Malik. "Efficient Querying of Distributed Provenance Stores". *the Proceeding of 2010 Challenges of Large Applications in Distributed Environments (CLADE)*, 2010.

[Ghem 03] S. Ghemawat, H. Gobioff, and S. Leung. "The Google File System". In: *19th ACM Symposium on Operating Systems Principles*, Oct. 2003.

[Gill 08] T. Gill, A. J. Gilliland, M. Whalen, and M. S. Woodley. "Introduction to Metadata". Online Edition Version 3.0, Getty Research Institute, 2008.

[Gobl 03] C. Goble, S. Pettifer, R. Stevens, and C. Greenhalgh. "Knowledge integration: in silico experiments in bioinformatics". In: I. Forster and C. Kesselman, Eds., *The Grid 2: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann, Nov. 2003.

[Gobl 05] C. Goble and S. Bechhoer. "OntoGrid: A Semantic Grid Reference Architecture". *CTWatchQUARTERLY*, Nov. 2005.

[Gobl 06] C. Goble. "S-OGSA: A Reference Semantic Grid Architecture". *the Proceeding of 16th Global Grid Forum*, 2006.

[Gold 05] S. Golder and B. A. Huberman. "The Structure of Collaborative Tagging Systems". *the Journal of Information Science*, 2005.

[Goul 10] C. M. Gould, F. Diella, A. Via, P. Puntervoll, C. Gemünd, S. Chabanis-Davidson, M. S., A. Sayadi, J. C. Bryne, C. Chica, M. Seiler, N. E. Davey, N. Haslam, R. J. Weatheritt, A. Budd, T. Hughes, J. Pas, L. Rychlewski, G. Travé, R. Aasland, M. Helmer-Citterich, R. Linding, and T. J. Gibson. "ELM: the status of the 2010 eukaryotic linear motif resource". *Nucleic Acids Research*, Vol. 38, Database issue, pp. D167–D180, 2010.

[Gray 07] J. Gray. "Jim Gray on eScience: A Transformed Scientific Method". *The Fourth Paradigm: Data-Intensive Scientific Discovery*, 2007.

[Gref 97] P. Grefen and J. Widom. "Protocols for Integrity Constraint Checking in Federated Databases". *Distributed and parallel databases*, Vol. 5, No. 4, pp. 327–355, 1997.

[Grie 10] S. Griep and U. Hobohm. "PDBselect 1992-2009 and PDBfilter-select". *Nucleic Acids Research*, Vol. 38 Database issue, pp. D318–D319, 2010.

[Gros 08] R. L. Grossman. "A Quick Introduction to Clouds". Tech. Rep., Open Data Group, 2008.

[Gros 09] R. L. Grossman, Y. Gu, and M. Sabala. "The Open Cloud Testbed: A Wide Area Testbed for Cloud Computing Utilizing High Performance Network Services". Tech. Rep., Open Cloud Consortium (OCC), 2009.

[Gros 98] D. Gross and C. M. Harris. *Fundamentals of Queuing Theory*. Wiley-Interscience, Feb. 1998.

[Grub 93] T. R. Gruber. "A Translation Approach to Portable Ontology Specifications". *the Knowledge Acquisition*, Vol. 5, No. 2, 1993.

[Gu 02] L. Gu, L. Budd, A. Caycl, C. Hendricks, M. Purnell, and C. Rigdon. "A Practical Guide to DB2 UDB Data Replication V8". *DB2 Redbooks*, Dec. 2002.

[Gu 09] Y. Gu and R. L. Grossman. "Sector and Sphere: the Design and Implementation of a High-Performance Data Cloud". *Philosophical Transactions of The Royal Society A*, 2009.

[Gudg 05] M. Gudgin and M. Hadley. "Web Services Addressing 1.0 – Core". Technical Reports, W3C, 2005.

[Gupt 93] A. Gupta and J. Widom. "Local Verification of Global Integrity Constraints in Distributed database". *ACM SIGMOD Record*, 1993.

[Gupt 94] A. Gupta, Y. Sagiv, J. D. Ullman, and J. Widom. "Constraint Checking with Partial Information". *the Proceeding of the 13th ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, 1994.

[Guti 09] M. E. Gutierrez, S. M. Pahlevi, O. Corcho, and I. Kojima. "Accessing RDF(S) Data Resources in Service-based Grid Infrastructures". *Concurrency and Computation: Practice and Experience*, Vol. 21, No. 8, pp. 1029–1051, Feb. 2009.

[Guzm 10] J. C. Guzman and B. Humpherys. "The Australian SKA Pathfinder (ASKAP) Software Architecture". In: *Proceedings of the SPIE*, 2010.

[Halp 07] H. Halpin, V. Robu, and H. Shepherd. "The Complex Dynamics of Collaborative Tagging". In: *the Proceeding of the 16th international conference on World Wide Web*, 2007.

[Han 11] L. Han, J. van Hemert, and R. Baldock. "Automatically Identifying and Annotating Mouse Embryo Gene Expression Patterns". *Oxford Journals: Bioinformatics (2011)*, Feb. 2011.

[Harm 09] T. Harmer, P. Wright, C. Cunningham, and R. Perrott. "Provider-Independent Use of the Cloud". *Euro-Par 2009 Parallel processing*, pp. 454–465, 2009.

[Hass 09] O. Hassanzadeh. "A Declarative Framework for Semantic Link Discovery over Relational Data". In: *18th World Wide Web Conference (WWW2009)*, 2009.

[Hay 09] T. Hay, S. Tansley, and K. Tolle. *The Fourth Paradigm: Data-Intensive Scientific Discovery*. Microsoft Research, 2009.

[Hosc 04] W. Hoschek. "The Colt Distribution: Open Source Libraries for High Performance Scientific and Technical Computing in Java". Tech. Rep., CERN, 2004.

[Hull 06] D. Hull, K. Wolstencorft, R. Stevens, C. Goble, M. R. Pocock, P. Li, and T. Oinn. "Taverna: a tool for building and running workflows of services". *Nucleic Acids Research*, Vol. 34 Web Service issue, No. doi:10.1093/nar/gkl320, 2006.

[ISO 08] ISO. "Digital Object Identifier (DOI) System". *ISO Standards*, Apr. 2008.

[ISOI 01] ISO/IEC. "ISO/IEC 11578:1996 Information Technology – Open Systems Interconnection –Remote Procedure Call". *ISO*, 2001.

[Jain 91] R. Jain. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation and Modeling. ISBN: 0471503361*, Wiley-Interscience, New York, NY, Apr. 1991.

[Jin 97] L. Jin and R. V. Lloyd. "In situ hybridization: methods and applications". *J Clin Lab Anal*, Vol. 11, No. 1, pp. 2–9, 1997.

[Jone 09] A. R. Jones, C. C. Overly, and S. M. Sunkin. "The Allen Brain Atlas: 5 years and beyond". *Nature Reviews Neuroscience 10*, No. doi:10.1038/nrn2722, pp. 821–828, Nov. 2009.

[Kava 01] M. Kavalec, V. Svatek, and P. Strossa. "Web Directories as Training Data for Automated Metadata Extraction". In: *the 2001 Semantic Web Mining, Workshop at ECML/PKDD*, 2001.

[Kell 09] S. Kelling, W. M. Hochachka, D. Fink, M. Riedewald, R. Caruana, G. Ballard, and G. Hooker. "Data Intensive Science: A New Paradigm for Biodiversity Studies". *BioScience*, Vol. 59, No. 7, pp. 613–620, Jul 2009.

[Kett 08] R. Kettimuthu, L. Wantao, F. Siebenlist, and I. Foster. "Communicating Security Assertions over the GridFTP Control Channel". In: *IEEE International Conference on e-Science*, Dec. 2008.

[Kim 02] S. K. Kim and D. Carrington. "A Formal Model of the UML Metamodel: The UML State Machine and Its Integrity Constraints". In: *the Proceeding of the 2nd International Conference of B and Z Users on Formal Specification and Development in Z and B*, 2002.

[Kim 06] J. Kim, Y. Gil, and V. Ratnakar. "Semantic Metadata Generation for Large Scientific Workflow". In: *the Proceeding of the 5th International Semantic Web Conference (ISWC)*, 2006.

[Kim 10] P. Kim, S. Yoon, N. Kim, S. Lee, M. Ko, H. Lee, H. Kang, J. Kim, and S. Lee. "ChimerDb2.0—a knowledgebase for fusion genes updated". *Nucleic Acids Research*, Vol. 38 Database issue, pp. D81–D85, 2010.

[Klem 03] A. Klemm, C. Lindemann, and M. Lohmann. "Modeling IP traffic using the batch Markovian arrival process". *Performance Evaluation*, Vol. 54, No. 2, pp. 149–173, 2003.

[Krek 04] J. Kreku, J. Penttila, J. Kangas, and P. Soininen. "Workload simulation method for evaluation of application feasibility in a mobile multiprocessor platform". *Proceeding of the EUROMICRO Systems on Digital System Design (DSD'04)*, 2004.

[Kuns 03] P. Kunszt, E. Laure, H. Stockinger, and K. Stockinger. "Data Grid Replica Management with Reptor". In: *the Proceedings of 5th international conference on parallel processing and applied mathematics*, 2003.

[Laks 08] A. Lakshman and P. Malik. "Cassandra: A Structured Sotrage System on a P2P Network". *SIGMOD*, 2008.

[Lame 02] H. Lamehamedi, B. Szymanski, Z. Shentu, and E. Deelman. "Data Replication Strategies in Grid Environments". In: *the Proceeding of the 5th International Conference on Algorithms and Architecture for Parallel Processing*, 2002.

[Lang 09] B. Langmead, C. Trapnell, M. Pop, and S. L. Salzberg. "Ultrafast and memory-efficient alignment of short DNA sequences to the human genome". *Genome Biol*, Vol. 10, No. R25, 2009.

[Leac 05] P. Leach, M. Mealling, and R. Salz. "A Universally Unique Identifier(UUID) URN Namespace". Standards, IETF, Jul. 2005.

[Lecl 88] C. Lecluse, P. Rtchard, and F. Velez. "O2, an Object-Oriented Data Model". *ACM SIGMOD Record*, Vol. 17, No. 3, 1988.

[LEcu 02] P. L'Ecuyer, L. Meliani, and J. Vaucher. "SSJ: A Framework For Stochastic Simulation in Java". In: *Proceeding of the 2002 Winter Simulation Conference*, 2002.

[Lees 10] J. Lees, C. Yeats, O. Redfern, A. Clegg, and C. Orengo. "Gene3D: merging structure and function for a Thousand genomes". *Nucleic Acids Research*, Vol. 38, Database issue, pp. D296–D300, 2010.

[Lein 10] R. Leinonen, R. Akhtar, E. Birney, J. Bonfield, L. Bower, M. Corbett, Y. Cheng, F. Demiralp, N. Faruque, N. Goodgame, R. Gibson, G. Hoad, C. Hunter, M. Jang, S. Leonard, Q. Lin, R. Lopez, M. Maguire, H. McWilliam, S. Plaister, R. Radhakrishnan, S. Sobhany, G. Slater, P. T. Hoopen, F. Valentin, R. Vaughan, V. Zalunin, D. Zerbino, and G. Cochrane. "Improvements to services at the European Nucleotide Archive". *Nucleic Acids Research*, Vol. 38, Database issue, pp. D39–D45, 2010.

[Li 07] H. Li, R. Heusdens, M. Heusdens, and L. Wolters. "Analysis and Synthesis of Pseudo-Periodic Job Arrivals in Grids: A Matching Pursuit Approach". In: *the Proceeding of 7th IEEE Intl. Sym. On Cluster Computing and the Grid (CCGrid07)*, May 2007.

[Li 92] W. Li. "Random Texts Exhibit Zipf's-Law-Like Word Frequency Distribution". *IEEE Transactions on Information Theory*, Vol. 36, No. 6, pp. 1842–1845, 1992.

[Life 05] "Life Sciences Identifiers Specification". Specification, OMG, 2005.

[Limi 05] L. Liming. "Large-Scale Data Replication for LIGO". *ClusterWorld submission*, Jul. 2005.

[Liu 08] X. Liu, J. Zhang, J. Zhang, and X. He. "Analyzing the Self-similarity of the TCP Bulk Flow". In: *the Proceeding of the 9th International Conference for Young Computer Scientists*, pp. 2718–2721, Nov. 2008.

[Loba 10] M. Y. Lobanov, B. A. Shoemaker, S. O. Garbuzynskiy, J. H. Fong, A. R. Panchenko, and O. V. Galzitskaya. "ComSin: database of protein structures in bound (complex) and unbound (single) states in relation to their intrinsic disorder". *Nucleic Acids Research*, Vol. 38, Database issue, pp. D283–D287, 2010.

[Lord 05] P. Lord, P. Alper, C. Wroe, and C. Goble. "Feta: A light-weight architecture for user oriented semantic service discovery". In: *the Proceeding of 2005 European Semantic Web Conference*, 2005.

[Magl 06] D. Maglott, J. Ostell, K. D. Pruitt, and T. Tatusova. "Entrez Gene: gene-centered information at NCB". *Nucleic Acids Research*, Vol. Special Issue on Synthetic Biology, No. doi:10.1093/nar/gk1993, 2006.

[Malg 06] H. Malgouyres and G. Motet. "A UML Model Consistency Verification Approach Based on Meta-model Formalisation". In: *the Proceedings of the 2006 ACM symposium of Applied computing*, 2006.

[Marj 09] U. Marjit and S. Jana. "Mashup: An Emerging Content Aggregation Web2.0 Paradigm". In: *7th International CALIBER-2009*, Feb. 2009.

[MCAT 10] MCAT. "MCAT". Tech. Rep., http://www.sdsc.edu/srb/index.php/MCAT, Lastchecked Dec. 2010.

[Meal 02] M. Mealling and R. Denenberg. "Report from the Join W3C/IETF URI Planning Interest Group: Uniform Resource Identifiers (URIs), URLs and Uniform Resource Names (URNs): Clarification and Recommendations". Standards, IETF, Aug. 2002.

[Meal 99] M. Mealling and R. Daniel. "URI Resolution Services Necessary for URN Resolution". Standards, IETF, Jan. 1999.

[Moor 06] R. Moore. "Building Preservation Environments with Data Grid Technology". *American Archivist*, Vol. 69, No. 1, pp. 139–158, Jul. 2006.

[Moor 08a] R. Moore. "Data Federation Network". Tech. Rep., Data Intensive Cyber Environments Group, 2008.

[Moor 08b] R. Moore and A. Rajasekar. "IRODS: Integrated Rule-Oriented Data System". Tech. Rep., IRODS White Paper, Sep. 2008.

[Moun 04] D. W. Mount. *Bioinformatics: Sequence and Genome Analysis*. Cold Spring Harbor Laboratory Press, 2004.

[Musc 05] L. Muscariello, M. Mellia, M. Meo, M. A. Marsan, and R. L. Cigno. "Markov Models of Internet Traffic and a New Hierarchical MMPP Model". *Computer Communication*, Vol. 28, pp. 1835–1851, 2005.

[Myer 03] J. D. Myers, C. Pancerella, C. Lansing, K. L. Schuchardt, and B. Didier. "Multi-scale science: supporting emerging practice with semantically derived provenance". In: *ISWC 2003 Workshop: Semantic Web Technologies for Searching and Retrieving Scientific Data*, 2003.

[NCRR 06] NCRR. "caBIG Overview". Tech. Rep., National Institutes of Health National Center for Research Resources, 2006.

[Nent 01] C. Nentwich, W. Emmerich, and A. Finkelstein. "Static Consistency Checking for Distributed Specifications". In: *the Proceedings of the 16th Automated Software Engineering Conference*, 2001.

[Nich 06] C. Nicholson. "Dynamic Data Replication in LCG 2008". In: *the 2006 UK e-Science All Hands Conference*, 2006.

[NISO 04] NISO. *Understanding Metadata*. NISO Press, National Information Standards Organization, 2004.

[Nurm 09] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov. "The Eucalyptus Open-Source Cloud-Computing System". In: *9th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID)*, pp. 124–131, 2009.

[OASI 04] OASIS. "Introduction to UDDI Important Features and Functional Concepts". White paper, OASIS, 2004.

[Ober 99] J. Oberg. "Why The Mars Probe Went Off Course". *IEEE Spectrum*, Vol. 36, 1999.

[Oinn 04] T. Oinn, M. Addis, J. Ferris, J. Marvin, M. Senger, M. Greenwood, T. Carver, K. Glover, M. Pocock, A. Wipat, and P. Li. "Taverna: a tool for the composition and enactment of bioinformatics workflows". *Bioinformatics*, Vol. 20, No. 17, 2004.

[Okam 03] M. Okamoto, T. Hanai, and C. Arima. "Gene Expression Analysis Using Fuzzy K-Means Clustering". *Genome Informatics*, Vol. 14, pp. 334–335, 2003.

[Olen 90] K. M. Olender and L. J. Osterweil. "Cecil: A Sequencing Constraint Language for Automatic Static Analysis Generation". *IEEE Transactions On Software Engineer*, Vol. 16, No. 3, 1990.

[Olst 08] C. Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins. "Pig Latin: A Not-So-Foreign Language for Data Processing". *Proceddings of the SIGMOD'08*, Jun. 2008.

[Pahl 08] S. M. Pahlevi and I. Kojima. "Semantic Grid Resource Monitoring and Discovery with Rule Processing based on the Time-series Statistical Data". In: *9th Grid Computing Conference*, 2008.

[Prli 07] A. Prlic, T. A. Down, E. Kulesha, R. D. Finn, A. Kahari, and T. Hubbard. "Integrating sequence and structural biology with DAS". *BMC Bioinformatics*, Vol. 8, No. 333, 2007.

[Raja 02a] A. Rajasekar, R. Marciano, and R. Moore. "Collection-Based Persistent Archives". In: *16th IEEE Symposium on Mass Storage Systems*, 2002.

[Raja 02b] A. Rajasekar and M. Wan. "SRB and SrbRack Components of a Virtual Data Grid Architecture". In: *the Proceeding of 2002 Advance Simulation Technologies Conference*, 2002.

[Raja 03] A. Rajasekar, M. Wan, R. Moore, W. Schroeder, G. Kremenek, A. Jagatheesan, C. Cowart, B. Zhu, S. Y. Chen, and R. Olschanowsky. "Storage Resource Broker - Managing Distributed Data in a Grid". *Computer Society of India Journal*, Vol. 33, No. 4, pp. 42–54, Oct. 2003.

[Rast 93] R. Rastogi, S. Mehrotra, Y. Breitbart, H. F. Korth, and A. Silberschatz. "On Correctness of Non-serializable Executions". In: *the Proceedings of the 12th ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, 1993.

[Ratt 10] T. Rattei, P. Tischler, S. Gotz, M. A. Jehl, J. Hoser, R. Arnold, A. Conesa, and H. W. Mewes. "SIMAP – a comprehensive database of pre-calculated protein sequence similarities, domains, annotations and clusters". *Nucleic Acids Research*, Vol. 38, Database issue, pp. D223–D226, 2010.

[Resn 01] Resnick. "Internet Message Format". Standards, IETF, 2001.

[Rhod 01] M. G. Rhode. "Integrating Semantics for Object-Oriented System Models". In: *the Proceeding of the 28th International Colloquium on Automata, Language and Programming*, 2001.

[Rour 10] D. de Roure, C. Goble, S. Aleksejevs, S. Bechhofer, J. Bhagat, D. Cruickshank, D. Hull, Y. Lin, D. Michaelides, D. Newman, and R. Procter. "Towards Open Science: The myExperiment approach". *CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE*, No. DOI 10.1002/cpe.1601, Jul 2010.

[Rusi 91] M. Rusinkiewicz, A. Sheth, and G. Karabatis. "Specifying Interdatabase Dependencies in a Multidatabase Environment". *Computer*, Vol. 24, pp. 46–53, 1991.

[Sala 06] L. Salayandia, Y. Huang, A. Q. Gates, and S. Roach. "GeoNet: Use of Grid Technologies in Geoinformatics for the Transition Zone between the Colorado Plat". In: A. K. Sinha, Ed., *Geoinformatics: Data to Knowledge (Special Paper)*, pp. 183–194, Geological Society of America, 2006.

[Sant 06a] N. Santos and B. Koblitz. "Distributed Metadata with the AMGA Metadata Catalog". In: *the Proceeding of the 2006 Workshop on Next-Generation Distributed Data Management (HPDC-15)*, 2006.

[Sant 06b] N. Santos and B. Koblitz. "Metadata service on the grid". *Nuclear Instruments & Methods in Physics Research A*, Vol. 559, pp. 53–56, 2006.

[Saye 10] E. W. Sayers, T. Barrett, D. A. Benson, E. Bolton, S. H. Bryant, K. Canese, V. Chetvernin, D. M. Church, M. DiCuccio, S. Federhen, M. Feolo, L. Y. Geer, W. Helmberg, Y. Kapustin, D. Landsman, D. J. Lipman, Z. Lu, T. L. Madden, T. Madej, D. R. Maglott, A. Marchler-Bauer, V. Miller, I. Mizrachi, J. Ostell, A. R. Panchenko, K. D. Pruitt, G. D. Schuler, E. Sequeira, S. T. Sherry, M. Shumway, K. Sirotkin, D. Slotta, A. Souvorov, G. Starchenko, T. Tatusova, L. Wagner, Y. Wang, W. J. Wilbur, E. Yaschenko, and J. Ye. "Database resources of the National Center for Biotechnology Information". *Nucleic Acids Research*, Vol. 38, Database issue, pp. D5–D16, 2010.

[Schl 05] S. D. Schlueter, M. D. Wilkerson, E. Huala, S. Y. Rhee, and V. Brendel. "Community-based gene structure annotation". *TRENDS in Plant Science*, Vol. 10, No. 1, Jan. 2005.

[Schu 96] G. D. Schuler, J. A. Epstein, H. Ohkawa, and J. A. Kans. "Entrez: Molecular Biology Database and Retrieval System". *Methods Enzymal*, Vol. 266, pp. 141–162, 1996.

[Shah 00] S. Shah-Heydari and T. Le-Ngoc. "MMPP Models for Multimedia Traffic". *Telecommunication System*, Vol. 15, pp. 273–293, Feb. 2000.

[Shar 02] R. Sharma. *Microsoft SQL Server 2000: A Guide to Enhancements and New Features*. Addison-Wesley Professional, 2002.

[Sigr 10] C. J. A. Sigrist, L. Cerutti, E. de Castro, P. S. Langendijk-Genevaux, V. Bulliard, A. Bairoch, and N. Hulo. "PROSITE, a protein domain database for functional characterization and annotation". *Nucleic Acids Research*, Vol. 38, Database issue, pp. D161–D166, 2010.

[Sigu 08] B. Sigurbjornsson and R. van Zwol. "Flickr Tag Recommendation based on Collective Knowledge". In: *WWW 2008*, Apr. 2008.

[Simm 05] Y. L. Simmhan, B. Plale, and D. Gannon. "A Survey of Data Provenance in e-Science". *the ACM SIGMOD Record*, Vol. 34, No. 3, Sep 2005.

[Sing 06] V. Singh, J. Gray, A. Thakar, A. S. Szalay, J. Raddick, B. Boroski, S. Lebedeva, and B. Yanny. "SkyServer Traffic Report – The First Five Years". Tech. Rep. MSR TR-2006-190, Microsoft, Dec. 2006.

[Sinn 07] R. O. Sinnott, G. Stewart, A. Asenov, C. Millar, G. Roy, S. Roy, and A. Brown. "Grid Infrastructures for the Electronics Domain: Requirements and Early Prototypes from an EPSRC Pilot Project". In: *the Proceeding of the UK e-Science All Hands Meeting*, 2007.

[Sobe 08] W. Sobel, S. Subramanyam, A. Sucharitakul, J. Nguyen, H. Wong, S. Patil, A. ando Fox, and D. Patterson. "Cloudstone: Multi-Platform, Multi-Language Benchmark and Measurement Tools for Web2.". In: *the First Workshop on Cloud Computing and its Applications (CCA)*, 2008.

[Soli 07] M. A. Soliman, I. F. IIyas, and K. C. Chang. "URank: Formulation and Efficient Evaluation of Top-k Queries in Uncertain Databases". In: *SIGMOD*, 2007.

[Soll 99] K. Sollins and L. Masinter. "Functional Requirements for Uniform Resource Name". Standards, IETF, 1999.

[Stan 04] I. Standard. "Information technology – Metadata Registries (MDR)". *ISO-IEC 11179, second edition*, 2004.

[Stra 03] R. Straeten, T. Men, J. Simmonds, and V. Jonckers. "Using Description Logic to Maintain Consistency between UML Models". *UML" 2003 – The Unified Modeling Language*, pp. 326–340, 2003.

[Szal 00a] A. S. Szalay, P. Z. Kunszt, A. Thakar, J. Gray, and D. Sluts. "Designing and Mining Multi-Terabyte Astronomy Archives: The Sloan Digital Sky Survey". *the Proceeding of the 2000 ACM SIGMOD*, 2000.

[Szal 00b] A. S. Szalay, P. Z. Kunszt, A. Thakar, J. Gray, D. Sluts, and R. J. Brunner. "Designing and Minding Multi-Terabyte Astronomy Archives: The Sloan Digital Sky Survey". In: *the Proceeding of the 2000 ACM SIGMOD International Conference on Management of Data*, 2000.

[Szal 09] A. S. Szalay, G. Bell, J. Vanderberg, A. Wonders, R. Burns, D. Fay, J. Heasley, T. Hey, M. Nieto-SantiSteban, A. Thakar, C. van Ingen, and R. Wilton. "GrayWulf: Scalable Cluster architecture for Data Intensive Computing". In: *the 42nd Hawaii International Conference of System Sciences*, 2009.

[Tamm 02] V. Tamma and T. B. Capon. "An Ontology Model to Facilitate Knowledge-Sharing in Multi-agent System". *the Knowledge Engineering Review*, Vol. 17, No. 1, 2002.

[Tuff 06] M. M. Tuffield, S. Harris, C. Brewster, N. Gibbins, F. Ciravegna, D. Sleeman, N. R. Shadbolt, and Y. Wilks. "Image annotation with photocopain". In: *the Proceeding of Semantic Web Annotation of Multimedia (SWAMM-06) Workshop at the World Wide Web Conference 06*, 2006.

[Vela 10] S. Velankar, C. Best, B. Beuth, H. Boutselakis, N. Cobley, A. W. Sousa Da Silva, D. Dimitropoulos, A. Golovin, M. Hirshberg, M. John, E. B. Krissinel, R. Newman, T. Oldfield, A. Pajon, C. J. Penkett, J. Pineda-Castillo, G. Sahni, S. Sen, R. Slowley, A. Suarez-Uruena, J. Swaminathan, G. van Ginkel, W. F. Vranken, K. Henrick, and G. J. Kleywegt. "PDBe:Protein Data Bank in Europe". *Nucleic Acids Research*, Vol. 38, Database issue, pp. D308–D317, 2010.

[Vise 04] A. Visel, C. Thaller, and G. Eichele. "GenePaint.org: an atlas of gene expression patterns in the mouse embryo". *Nucleic Acids Research*, Vol. 32, Database issue, pp. D552–D556, 2004.

[Volz 09] J. Volz, C. Bizer, M. Gaedke, and G. Kobilarov. "Silk – A Link Discovery Framework for the Web of Data". In: *the 2nd workshop on Linked Data on the Web (LDOW2009)*, 2009.

[Voss 07] J. Voss. "Tagging, Folksonomy and Co-Renaissance of Manual Indexing?". In: *the Proceeding of the 10th International Symposium of Information Science*, 2007.

[Voss 95] G. Vossen. "On Formal Models for Object-Oriented Database". *the ACM SIGPLAN OOPS Messenger*, Vol. 6, No. 1, 1995.

[W3C 01] W3C. "URIs, URLs and URNs: Clarifications and Recommendations 1.0". Technical Reports, W3C, URI Planning Interest Group, Sep. 2001.

[W3C 04] W3C. "RDF/XML Syntax Specification". white paper, W3C, 2004.

[Wal 07] T. W. Wal. "Folksonomy Coinage and Definition". Tech. Rep., http://www.vanderwal.net/folksonomy.html, 2007.

[Wang 07] Y. Wang, K. J. Addess, L. Y. Geer, J. He, S. He, S. Lu, T. Madej, A. Marchler-Bauer, A. P. Thiessen, N. Zhang, and S. H. Bryant. "MMDB: annotating protein sequences with Entrez's 3D-structure database". *Nucleic Acids Research*, Vol. 35, pp. D298–D300, 2007.

[Wang 10] Y. Wang, E. Bolton, S. Dracheva, K. Karapetyan, B. A. Shoemaker, T. O. Suzek, J. Wang, J. Xiao, J. Zhang, and S. H. Bryant. "An overview of the PubChem BioAssay resource". *Nucleic Acids Research*, Vol. 38, Database issue, pp. D255–D266, 2010.

[Weil 06] S. A. Weil, S. A. Brandt, E. L. Miller, D. D. E. Long, and C. Maltzahm. "Ceph: a scalable, high-performance distributed file system". In: *the Proceeding of the 7th symposium on Operating system design and implementation*, 2006.

[Wilk 10] P. Wilkinson, J. Sengerova, R. Matteoni, C. K. Chen, G. Soulat, A. Ureta-Vidal, S. Fessele, M. Hagn, M. Massimi, K. Pickford, R. H. Butler, S. Marschall, A. M. Mallon, A. Pickard, M. Raspa, F. Scavizzi, M. Fray, V. Larrigaldie, J. Leyritz, E. Birney, G. P. Tocchini-Valentini, S. Brown, Y. Herault, L. Montoliu, M. H. de Angelis, and D. Smedley. "EMMA—mouse mutant resources for the international scientific community". *Nucleic Acids Research*, Vol. 38, Database issue, pp. D570–D576, 2010.

[Yama 10] R. Yamashita, H. Wakaguri, S. Sugana, Y. Suzuki, and K. Nakai. "DBTSS provides a tissue specific dynamic view of Transcription Start Sites". *Nucleic Acids Research*, Vol. 38, Database issue, pp. D98–D104, 2010.

[Yang 10] J. H. Yang, P. Shao, H. Zhou, Y. Q. Chen, and L. H. Qu. "deepBase: a database for deeply annotating and mining deep sequencing data". *Nucleic Acids Research*, Vol. 38 Databse issue, pp. D123–D130, 2010.

[Zawo 04] J. D. Zawodny and D. J. Balling. *High Performance MySQL: Optimization, Backups, Replication and Load Balancing*. O'REILLY, 2004.

Appendix A

The Testing RDF Triples

The following RDF triples are used for the evaluation of SPARQL queries presented in section 6.5.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
       xmlns:binding="http://www.bindingstore.org/bindingrepresentation#">
   <binding:id rdf:about="http://www.bindingstore.org/bindings/73b2a8ee-dd76-47eb-962c-08f5430f313d">
       <binding:subject rdf:resource="http://www.dgemap.org/images/embryo029.png"/>
       <binding:object rdf:resource="http://www.dgemap.org/annotations/query?embryoId=029"/>
       <binding:tag>ISembryo029</binding:tag>
       <binding:tag rdf:nodeID="_tag1"/>
   </binding:id>
   <rdf:Description rdf:nodeID="_tag1">
       <binding:key>creationtime</binding:key>
       <binding:value rdf:datatype="http://www.w3.org/2001/XMLSchema#date">2010-02-11</binding:value>
   </rdf:Description>
   <binding:id rdf:about="http://www.bindingstore.org/bindings/43b2a8ee-dd76-47eb-962c-08f5430f313d">
       <binding:subject rdf:resource="http://www.dgemap.org/images/embryo030.png"/>
       <binding:object rdf:resource="http://www.dgemap.org/annotations/query?disorder=yes"/>
       <binding:tag rdf:nodeID="_tag2"/>
       <binding:tag>disordered</binding:tag>
       <binding:tag rdf:nodeID="_tag3"/>
   </binding:id>
   <rdf:Description rdf:nodeID="_tag2">
       <binding:key>creationtime</binding:key>
       <binding:value rdf:datatype="http://www.w3.org/2001/XMLSchema#date">2010-02-10</binding:value>
   </rdf:Description>
   <rdf:Description rdf:nodeID="_tag3">
       <binding:key>user<binding:key>
       <binding:value rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Bob</binding:value>
   </rdf:Description>
   <binding:id rdf:about="http://www.bindingstore.org/bindings/53b2a8ee-dd76-47eb-962c-08f5430f313a">
       <binding:subject rdf:resource="http://www.dgemap.org/images/embryo031.png"/>
       <binding:object/>
       <binding:tag rdf:nodeID="_tag4"/>
   </binding:id>
```

```
    <rdf:Description rdf:nodeID="_tag4">
        <binding:key>creationtime</binding:key>
        <binding:value rdf:datatype="http://www.w3.org/2001/XMLSchema#date">2010-02-21</binding:value>
    </rdf:Description>
    <binding:id rdf:about="http://www.bindingstore.org/bindings/13b2a8ee-dd76-47eb-962c-08f5430f313a">
        <binding:subject rdf:resource="http://www.dgemap.org/images/embryo030.png"/>
        <binding:object rdf:resource="http://www.dgemap.org/annotations/query?embryoId=030"/>
        <binding:tag>embryo</binding:tag>
        <binding:tag>annotated by HDBR</binding:tag>
        <binding:tag>head</binding:tag>
        <binding:tag rdf:nodeID="_tag5"/>
        <binding:tag rdf:nodeID="_tag6"/>
    </binding:id>
    <rdf:Description rdf:nodeID="_tag5">
        <binding:key>creationtime</binding:key>
        <binding:value rdf:datatype="http://www.w3.org/2001/XMLSchema#date">2010-02-21</binding:value>
    </rdf:Description>
    <rdf:Description rdf:nodeID="_tag6">
        <binding:key>user</binding:key>
        <binding:value rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Alison</binding:value>
    </rdf:Description>
</rdf:RDF>
```

Appendix B

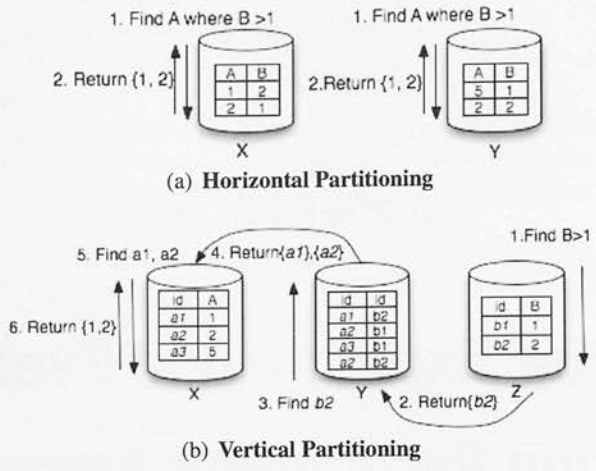
A Plan for Experimental Evaluation of Binding Partitioning Approaches in the Cloud

B.1 Motivation

The positive experiences with the Hadoop-based binding system lead to a proposal of new partitioning approaches for a binding Cloud.

Data partitioning splits the datasets into separate blocks of datasets, and places them on the storage servers. The ability to partition data in an efficient and high-performance manner requires a high level of skill. Automating this process remains an elusive goal. An appropriate partitioning can dramatically improve the system performance. In contrast, an inappropriate partitioning can actually decrease the performance even with the addition of more servers. Consider the example shown in Figure B.1, (a) illustrates a horizontal partitioning which splits rows into two stores X and Y, and (b) shows a vertical partitioning which splits fields and places them on three servers X, Y, and Z. Given a query "*Find A where B>1*", approach (a) allows two servers working simultaneously, and each of them processes half of the datasets. Total speed gain will be 50% (in theory). In the case of (b), the selection of "*B>1*" is performed on Z which ships the results to the server Y. Y filters out the datasets and passes them to X, and the latter scans the local data store and returns the final results. In this case, parallelisation cannot be performed by (b), and the data shipping appears to be the bottleneck. However, approach (b) will be very efficient for other types of query such as "*sum up all B and all A*".

The Cloud model provides a programmable interface, which makes it possible to send the functions to each slave server for executions. This can be called as *Computing Partitioning*, which, in combination of *Data Partitioning*, could offer a greater degree of flexibility.



(a) **Horizontal Partitioning**

(b) **Vertical Partitioning**

Figure B.1: Two Partitioning Approaches

Partitioning is a crucial for optimising the performance of the Cloud-based binding system. It relates to various computing processes, such as staging of data into computations, mapping of complex query onto resource nodes, redistribution of individual dataset, and replication for availability or durability.

B.2 Hypothesis

We hypothesise that a Cloud-based binding storage system with an intelligent partitioning mechanism is both scalable and efficient.

To test the hypotheses, novel partitioning approaches will be developed, and performance of the different approaches will be compared by controlled experiments. A simulation environment will be established. The current workload simulator will be extended so as to generate ultra-scale concurrent user requests. It is also possible to use existing Cloud workload simulation tool such as CloudStone [Sobe 08]. If the new binding service can support ultra scale synthetic workloads simulated via the load generators and present graceful performance features in efficiency and productivity, the project could be considered to have succeeded.

B.3 Partitioning Approaches

Three partitioning approaches will be tested in a Clound-based binding storage system.

A. Simple Random Partitioning. Initially, a Simple Random Partitioning approach will be practised, which uses hashing functions randomly distributing data among all nodes of a Cloud cluster.

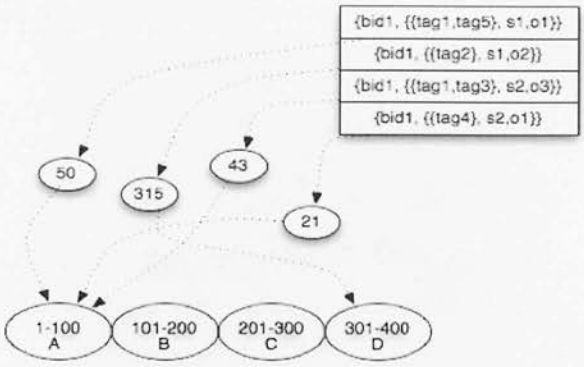


Figure B.2: Hash Based Simple Random Partitioning Algorithms

Recall the binding data structure, which can be easily converted into a Key/Value pair used by Map/Reduce programming, thus have:

$$< id, d, m, \{a_0, a_1, \ldots, a_n\} >$$

For a n-node Cloud cluster, given a N-key space, each node can be regarded as a hash bucket maintaining a partial key space [(x-1)×(**N/n**)+1, x×**N/n**]. On binding creation, each binding can be 'hashed'into an appropriate binding-bucket. Figure B.2 illustrates this process with a 4-node cluster and a 400-key space.

On binding searching, only buckets containing the relevant values will be scanned. For example, for a query "*Find bindings where 187<id<215*", B and C will parallelised process the query and deliver the results.

The Simple Random Partitioning will establish a baseline for a further study.

A novel partitioning methods is proposed here which uses Clustering algorithms. Clustering is the assignment of a set of observations into subsets (*clusters*) so that observations in the same *cluster* are similar in some sense. Bindings are quantised into *clusters* based on similarity, and the *clusters* are assigned to storage-nodes in the Cloud. Intuitively, if similar bindings are stored together, data locality will be high – if a binding satisfies a query, it is likely that the 'friends' of the binding will as well.

Two Clustering algorithms are considered: *K-Mean Clustering* and *Fuzzy K-Mean Clustering*.

B. K-Mean Clustering Partitioning. K-Mean Clustering is a widely used cluster formulation. Figure B.3 depicts the proposed K-Mean Clustering Partitioning. The algorithm works as follows: (1) It begins with a decision on the value of K, the number of *clusters* = the number of nodes. Then, (2) an initial partition is performed that classifies the bindings into K *clusters*. This can be done by using the Simple Random Partitioning approach described above. (3) Take each binding in sequence and compute its distance from the

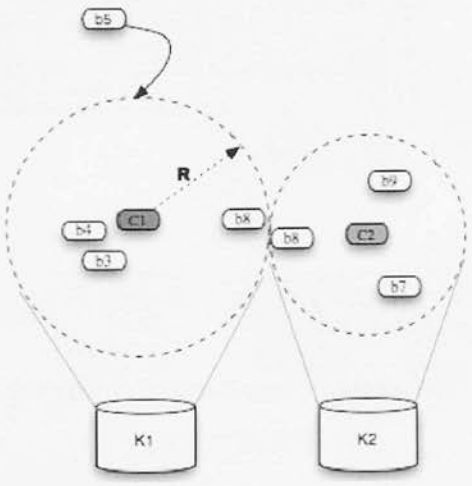


Figure B.3: A K-Mean Clustering Partitioning Approach

centroid of each of the *clusters*. The *centroid* can be a randomly chosen binding from the cluster. (4) If a binding is not currently in the cluster with the closest *centroid*, move this binding to that node and update the *centroid* of the cluster gaining the new binding and the cluster losing the binding. Finally, (5) repeat step (4) until convergence is achieved.

The *distance* between the binding and the *coentroid* is measured by similarity. The binding similarity function is defined as follows:

$$Sim(b_i, b_j) = w_o \times Sim(o_i, o_j) + w_s \times Sim(s_i, s_j) + w_T \times Sim(T_i, T_j) \qquad \text{(B.1)}$$

w_o, w_s, w_T are the weights of binding object, binding subject, and binding tags, respectively, while $Sim(o_i, o_j)$, $Sim(s_i, s_j)$, $Sim(T_i, T_j)$ are similarity functions for the three binding attributes. For the similarity functions for binding object and binding subject, the *distance* between URIs will be measured. For the similarity function for binding tag sets, two algorithms have been implemented, based on Jaccard's Coefficient[1] and based on the Vector Space Model[2].

C. Fuzzy K-Means Clustering. In K-Mean Clustering approach, the binding *clusters* are mutually exclusive, and a binding can only belong to exactly one *cluster*. Depending on the similarity function applied, a binding can be actually assigned to different *clusters*. The Fuzzy K-Mean Clustering partitioning is introduced here, which allows overlap of *clusters*. It will be interesting to see whether a certain level of redundancy could improve or degrade the system performance.

[1] See related material and documents.
[2] See related material and documents.

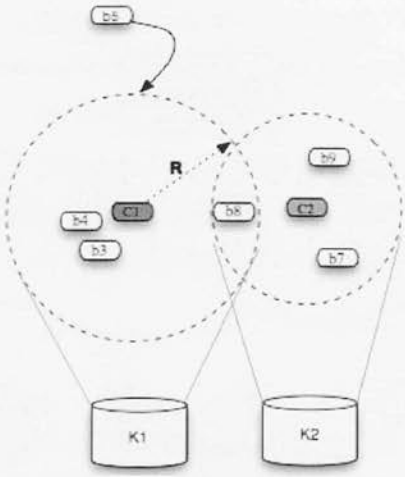


Figure B.4: A Fuzzy K-Mean Clustering Partitioning Approach

The equation used by the Fuzzy K-Mean Clustering is give as follows: [Bezd 81, Demb 03, Okam 03]:

$$J(K,m) = \sum_{k=1}^{K}\sum_{i=1}^{N}(u_{k1})^m d^2(x_i,c_k) \tag{B.2}$$

K and N are the number of *clusters* and bindings, m is a parameter which relates to 'fuzziness'of resulting *clusters*, u_{ki} is the degree of membership of binding x_i in *cluster* k, $d^2(x_i,c_k)$ is the *distance* from binding x_i to centroid c_k. The same similarity function developed above will still be applied. The parameters in the equation are the *cluster centroid* vector c_k and the components of the membership vectors u_{ki}. Calculated u_{ki} shows the belonging ratio to a *cluster* k, and *centroid* c_k shows the representative binding profile of a *cluster* k. These unknown parameters can be obtained using the following algorithm [Bezd 81, Demb 03]:

(1) Randomly select K bindings as initial *cenrioids* c_k, and partitions of all bindings around these *centroids*.

(2) For each *cluster*, compute the *centriodes* c_k:

$$c_k = \frac{\sum_{i=1}^{N}(u_{ki})^m x_i}{\sum_{i=1}^{N}(u_{ki})^m}; k = 1,2,\ldots K \tag{B.3}$$

(3) For each *cluster*, sequence scan the bindings and compute the membership values u_{ki}:

$$u_{ki} = \frac{1}{\sum_{i=1}^{K}\left[\frac{d^2(x_i,c_k)}{d^2(x_i,c_i)}\right]^{\frac{1}{m-1}}} \tag{B.4}$$

(4) Move the non-member bindings to a possible cluster, and repeat (2) and (3) until stabilisation.

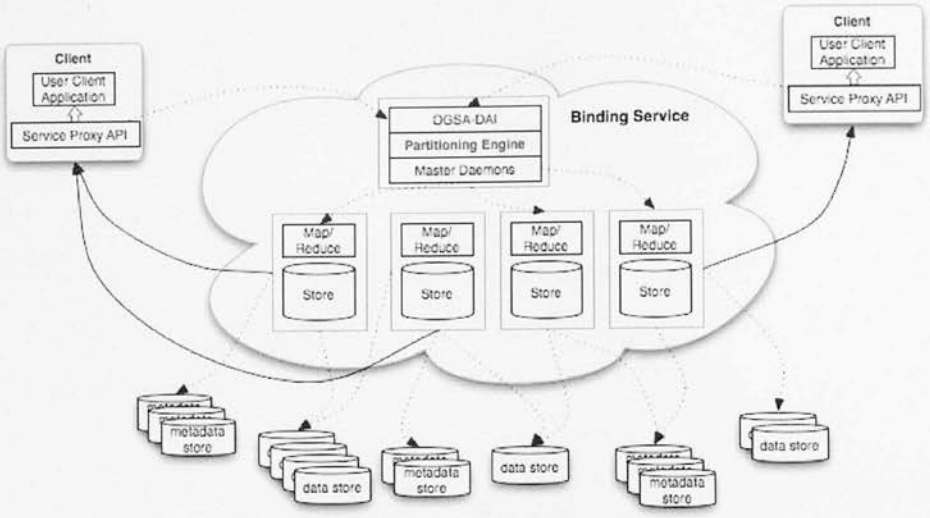


Figure B.5: A Cloud-based binding service

The proposed Clustering partitioning approach can be feasible and widely applicable since many scientific data analysis have to search uncertain data [Ge 08, Soli 07], where equality matching is no longer utilised, instead, similarity-based queries, top-k ranking or approximated answers become more popular. By using the Cloud computing, the computing can be highly efficient, since all nodes can process the calculations simultaneously. This will suit for classifying very large-scale data.

B.4 Experiment

The system architecture is depicted in Figure B.5. A Partitioning Engine will be developed by using Map/Reduce programming model. The web-based binding service will be glued on the Cloud programming interface, and be situated above the Partitioning Engine. The Partitioning Engine will be placed within the master node to interact with the Cloud operating system interface. The Hadoop will be used as the Cloud implementation framework.

Experiments will be established to compare the efficiency of the three partitioning approaches and evaluate the binding system performance in term of scalability and speed.

To compare the efficiency of the three partitioning algorithms, ≤20 typical queries will be implemented. It may not be able to evaluate 20 queries, but characteristic queries, such as point queries, range queries, similarity queries, top-k ranking, will be designed and implemented. Given different type of queries, for a fix number of the nodes, the (worst) average query response time and system throughputs will be measured and compared.

To evaluate the binding scalability, the system behaviours under increased workloads will be observed. It is expected that by adding the Cloud nodes, the system performance

will keep unchanged as the user numbers and access volumes increases. The workload simulator will be extended to generate requested workloads. The effects of three factors on the performance will be examined: (1) the types of workload, (2) the scale of the workload, and (3) the scale of the user.

Each configuration will be repeated 10 times, and the mean values, the Standard Errors (*SEs*) and the 95% Confidence Intervals (95% *CIs*) will be calculated. The experimental data will be presented, and useful conclusions will be drawn from carefully analysis.

B.5 Conclusion

The Cloud computing opens a new direction for the study of bindings and leads the research into a new distributed paradigm. Many interesting research can be done, and many open areas remain unexplored. It can be expected that some of the binding issues, i.e. performance and scalability, can probably be resolved, meanwhile new issues will arise which are challenging and exciting.

Jersey Public Library - Reference Library, Halkett Place, St Helier, Jersey JE2 4WH Production Order No. **503D**
A/c Number JER002CH

JE	Class	Author	Title	Date	Estimate	
70213054	C364.1523	Simons, S & Lithou, C	Murder in the Islands		£10.76	8" P/bk
02499789	C623.1	Partridge, Colin	Hitler's Atlantic Wall	1976	£10.76	10" P/bk
30039436	C623.8929	White, Capt Martin	Remarks on the Winds, the Tides & the Current of the C	1844	£138.56	Leather reback
30039436	C623.8929	White, Capt Martin	Sailing Directions for the English Channel....	1846	£131.60	Leather reback
70270902	C796.51		The Channel Island Way	2011	£13.16	8" P/Bbk
00850586	C821 MET	Metvier, George	Poesies Guernesiaises et Francaises	1883	£97.94	Leather reback
30038944	C900	Stevens, Charles	The Lettres Closes and Ancient Petitions 1200-1454 A Survey	1978	£20.60	12" P/bk
00850039	C911		The New Guide Book for Jersey and Guensey	[1842]	£41.90	7" Ant. Cloth
7021753X	J310 LSF		Denombrement des Habitantes de L'isle de Jersey, avec les noms de ceux de la paroisse de St Laurens	1788	£54.80	14" Ant. Cloth Rebind
01483048	J341.37		An Index of International agreements of interest to Jersey	[1997]	£20.60	12" P/bk
01720228	J385	Bonsor,N.R.P.	The Jersey Railway	1969	£13.46	9" P/bk
01471309	J720	Ferrari, Andre	Jerse's Lost Heritage	1996	£20.30	12" P/bk
30566576	J726.5	Easterbrook, Caroline	How does the glasswork of Rene Lalique in St Matthew's Church, Millbrook, Jersey, relate to his complete oeuvre?	1975	£20.60	12" P/bk
70057133	J730		The Jersey Public Art strategy		£17.90	12" P/bk
0249549X	J782.24	Larbalestier, P.G.	Emmanuel: a sacred cantata		£17.90	11" P/bk
70227489	J793.73	Henwood, R.C.	So you think you know Jersey?	1998	£10.76	9" P/bk
70227624	J793.73	Henwood, R.C.	So you think you know Jersey?	1998	£10.76	9" P/bk
70227705	J793.73	Henwood, R.C.	So you think you know Jersey?	1998	£10.76	9" P/bk
70227527	J793.73	Henwood, R.C.	So you think you know Jersey?	1998	£10.76	9" P/bk
02495813	J821 THO	Thompson, Vincent	St Helier, the hermit; a poetic vision	1834	£64.52	Cloth reback
70217491	J902		Local historical chronology	1892	£39.56	7" Ant. Cloth
01191888	J911	Black, C.B.	Guide to Jersey	1902	£31.58	7" Ant. Cloth
02203227	J911		A guide to Jersey with some account of its government, its laws and privileges	1856	£34.82	7" Ant. Cloth
0220326X	J911		A week's visit to Jersey	184-	£30.80	7" Ant. Cloth
02203383	J911		New and improved guide and visitor's souvenir of the Island of Jersey, containing excursions to its beautiful scenery...		£30.08	7" Ant. Cloth
70283192	J912 LSF	Stead, John	Map of Jersey [from Caesarea]	1799	£39.51	Repair & Encap
30634822	J914.234	Chamber of Commerce	Jersey Channel Islands	[1957]	£15.86	9" P/bk
0119190X	J914.234	States of Jersey Tourism	Official guide Jersey Channel Islands	[1947]	£13.46	9" P/bk
01332732	J92 FAL	Sutherland, Lucie	How did Philip Falle's career give him the opportunity to amass his unique library collection	[1990]	£20.60	12" P/bk
40124320	J929.2		Norrey's Inventaire d'une Bibliotheque de famile historique, heraldique...		£20.80	11" Cloth
01961322	J929.4	Stevens, Charles	A Catalogue of Jersey surnames	1970	£20.72	12" P/bk
40113841	J942.34	Lake, Chris	Images of the past	1987	£13.46	10" Scenic
00815764	J942.34	Lake, Chris	Images of the past	1987	£13.46	10" Scenic
70228868	J942.34084	Ginns, Michael	The German armed forces in Jersey 1940-45	2009	£17.90	11" P/bk
7022885X	J942.34084	Ginns, Michael	The German armed forces in Jersey 1940-45	2009	£17.90	11" P/bk
70022445	J942.34084	Jersey war tunnels	Ho8 German underground hospital		£15.86	9" P/bk
02491982	J942.34084	Langstaff-Ellis, J.W.	An examiniation of the degree And types of resistance displayed by the Islanders of Jersey to the German military authority during the years of occupation 1940-1945	1979	£20.72	12" P/bk
7022031X	J942.34084	Lewis, John	A doctor's occupation	n.d.	£10.76	7" P/bk
70220352	J942.34084	Lewis, John	A doctor's occupation	n.d.	£10.76	7" P/bk
7022028X	J942.34084	Lewis, John	A doctor's occupation	n.d.	£10.76	7" P/bk
	Periodical		Channel Islands Occupation Review No 38	2010	£12.66	9" P/bk
				Total Est.	**£1,180.43**	

Estimate supplied to Heather Morton 4.10.11 by Charles Dunn
Official order number:

GENERAL INSTRUCTIONS
Transfer old barcodes where necessary
Letter spines where necessary including classmarks and publication dates when requested
Photocopy inside front & rear covers if necessary and bind in or paste down on new pastedowns
Repair covers or pages if torn or needing patches
Retain sewing or perfect bind or resew if antiquarian and loose pages
Replace library sleeves on sleeved hardbacks
Retain and remount outer cover title panels where requested
Retain leather bindings and repair by rebacking with corner tips to boards if needed